INSTITUTO SUPERIOR DE TECNOLOGIAS AVANÇADAS

VIRTUALIZAÇÃO DE SERVIDORES: APLICAÇÃO NO DESENVOLVIMENTO DE UMA BIBLIOTECA VIRTUAL

LICENCIATURA INFORMÁTICA – ISTEC LISBOA

RELATÓRIO PARA A OBTENÇÃO DO GRAU DE LICENCIADO EM INFORMÁTICA

2015/2016

Aluno:

Nome: Luís António Martins Louro

Número: 1936

Orientador do Projeto:

Professor Doutor Pedro Brandão

Lisboa 2016

Índice

Dedicatória

Dedico este trabalho à minha família e a todos os que me apoiaram.

Agradecimentos

Agradeço aos meus pais, aos meus irmãos e ao meu cunhado pelo tempo disponibilizado nesta etapa da minha vida.

Agradeço, também, aos meus colegas de curso que sempre se disponibilizaram para me dar apoio nas dúvidas que surgiram ao longo da elaboração deste trabalho.

Por último, agradeço ao Professor Doutor Pedro Brandão por toda a orientação e incentivo para que este trabalho se pudesse realizar da forma que hoje o apresento.

Resumo

No padrão atual, com a elevada procura de equipamentos, aplicativos e sistemas operacionais, exigem-se máquinas cada vez mais poderosas. A necessidade de reduzir os custos sem perder serviços, exigiu a necessidade de consolidar várias máquinas numa só, um conceito denominado de virtualização. A virtualização é um método que permite ao utilizador instalar e utilizar mais do que um sistema operativo no mesmo computador, o que permite aumentar a produtividade do ambiente da TI (Tecnologia da Informação) com baixos custos. Existem vários *softwares* de virtualização, sendo que o Hyper-V se apresenta como a solução mais utilizada.

Em associação à elevada demanda de eficiência a baixo custo pela virtualização de sistemas, somos confrontados com a necessidade de uma gestão e acesso continuado a múltiplas bases de dados, encontradas na maioria dos documentos e páginas de internet, e que são coletâneas de informação armazenadas e que podem ser acedidas por vários utilizadores utilizando diferentes parâmetros. A ADO.NET *(ActiveX Data Objects.NET)*, introduzida em 2000 pela Microsoft, é uma tecnologia amigável usada para aceder, manipular e tratar vários tipos de dados relacionais, documentos XML e dados de aplicações. O SQL *(Structured Query Language)* é uma linguagem padrão para gestão e manipulação de dados relacionais através de SGBDs (Sistemas de Gestão de Bases de Dados) que permite trabalhar com várias bases de dados, tais como Access, SQL Server, Oracle, MySql, entre outras.

Durante este estado de arte, será explorado o conceito de virtualização e uma das plataformas associadas a esta metodologia, o Hyper-V. Serão também apresentadas e discutidas as características principais e respetivos componentes da tecnologia ADO.NET e da plataforma SQL Server 2014, que permitem a gestão de dados, o que implica a definição não só das estruturas em que irão ser armazenados, mas também os mecanismos para poder manipular os mesmos.

Palavras-chave: Virtualização, Hyper-V, ADO.NET, SQL Server 2014, Base de Dados

Abstract

In the current standard, with the high demand for equipment, applications and operating systems, is mandatory the demand for even more powerful machines. The need to reduce costs without losing services has urged the need to consolidate several machines in one, a concept called virtualization. Virtualization is a method that allows the user to install and use more than one operating system on the same computer, which allows productivity increase of the IT (Information Technology) environment, at a very low cost. Nowadays, we have available several software programs for virtualization, however Hyper-V has been one of the most used systems.

In addition to high low-cost efficiency demand for virtualization systems, we are faced with the need for management and continuous access to multiple databases, found in most documents and web pages. Databases consist on stored information collections that can be accessed by multiple users using different parameters. ADO.NET (ActiveX Data Objects.NET), introduced in 2000 by Microsoft, is a friendly technology used to access, manipulate and treat several different types of relational data, XML documents, and applications. SQL (Structured Query Language) is a standard language for managing and manipulating relational data using DBMS (Database Management System), which allows working with multiple databases such as Access, SQL Server, Oracle, MySql, among others.

During the present state-of-the-art, the concept of virtualization as well as one of the platforms associated with this methodology, Hyper-V, will be exploited. In addition, it will also be presented and discussed the main components from the data management technologies ADO.NET and SQL Server 2014 what implies not only the definition of the structures that will be stored, but also the mechanisms to manipulate those data.

Keywords: Virtualization, Hyper-V, ADO.NET, SQL Server 2014, Database

Lista de abreviaturas

SO - Sistema Operativo

CPU - *Central Processing Unit*

OU - *Organizational Unit*

GPO - *Group Policy Object*

AD - *Active Directory*

VMM - *Virtual Machine Manager*

SCVMM - *System Center Virtual Machine Manager*

PDF - *Portable Document Format*

NAT - *Network address translation*

API - *Application Programming Interface*

IOMMU - *Input–output memory management unit*

ANSI - *American National Standards Institute*

PC - *Personal Computer*

BD – *Database*

ISO - Organização Internacional de Normalização

XML – *eXtensible Markup Language*

SQL – *Structured Query Language*

SGBD – Sistema de Gerenciamento de Banco de Dados

TI – Tecnologias de Informação

Lista de figuras

Lista de tabelas

Estado da Arte

1. Introdução

Os maiores desafios encontrados atualmente pelos responsáveis pela gestão da Tecnologia de Informação (TI) de uma empresa são: utilização rentável das infraestruturas tecnológicas, capacidade de dar resposta a novos desafios empresariais e flexibilidade na adaptação às mudanças organizacionais. Associado à necessidade de resposta a estas situações, aliam-se as restrições orçamentais e exigências regulatórias cada vez mais rigorosas. Neste contexto, a virtualização surge como uma tecnologia fundamental que consegue dar resposta a estes desafios impostos aos gestores informáticos de forma criativa e inovadora.

O desenvolvimento da virtualização de um sistema é o resultado de cerca de 40 anos de investigação (Fuchi, Tanaka, Manago, & Yuba, 1969; Rosin, 1969), sendo que os primórdios da sua fundação remontam ao início dos anos 70 (Goldberg R. P., 1973; Goldberg R. P., 1974). Contudo, a última década assistiu a uma explosão no desenvolvimento da metodologia e aplicações da virtualização (Pearce, Zeadally, & Hunt, 2013) devido às inúmeras vantagens que a sua implementação tem demonstrado na gestão das TI.

2. Virtualização

2.1. História da Virtualização

A virtualização teve a sua primeira referência num artigo publicado na Conferência Internacional de Processamento de Informação, intitulado "*Time sharing processing in large fast computers*", em Nova York, no decorrer do ano de 1959. Neste artigo, Christopher Strachey debruçava-se sobre a multiprogramação em tempo partilhado, definindo novas formas de utilização de máquinas de grande capacidade e os seus respetivos recursos de *hardware* (Veras & Kassick, 2011). No seguimento do seu trabalho, o *Massachusetts Institute of Technology* (MIT) desenvolveu o *Compatible Time Sharing System* (CTSS), cujo padrão foi utilizado por fabricantes de máquinas de grande capacidade.

A virtualização foi pela primeira vez implementada pela IBM por volta da década de 70, de forma a particionar os computadores do *mainframe* em máquinas virtuais separadas (Creasy, 1981). Estas partições possibilitavam que os *mainframes* assumissem várias tarefas em simultâneo. A máquina principal era um IBM 704 (M44) e cada máquina virtual era uma imagem experimental da máquina principal (44X). O espaço de endereçamento do 44X residia na hierarquia de memória do M44 e era utilizado um sistema de memória virtual e multiprogramação. Esta realidade foi largamente superada entre os anos de 70 a 80 com o aparecimento dos computadores pessoais que motivaram o abandono da virtualização (Mattos & Perales, 2008). Só mais tarde, no fim dos anos 90 é que a virtualização dos sistemas operativos voltou a ser utilizada e redimensionada até alcançar a dimensão total que apresenta atualmente.

2.2. Definição

A virtualização é uma tecnologia que permite dividir os recursos de uma única máquina de elevada performance, velocidade e armazenamento em vários sistemas operacionais, no mesmo local físico ou *hardware* (Laureano, 2006). Os principais objetivos do uso da virtualização, segundo Kusnetzky são a escalabilidade,

confiabilidade, agilidade, disponibilidade, altos níveis de performance e segurança centralizada (Kusnetzky, 2011).

2.3. Estrutura interna

A infraestrutura virtual permite criar uma camada de abstração entre a parte da computação, do armazenamento de dados e redes, e as aplicações que se encontram em funcionamento (Figura 1).

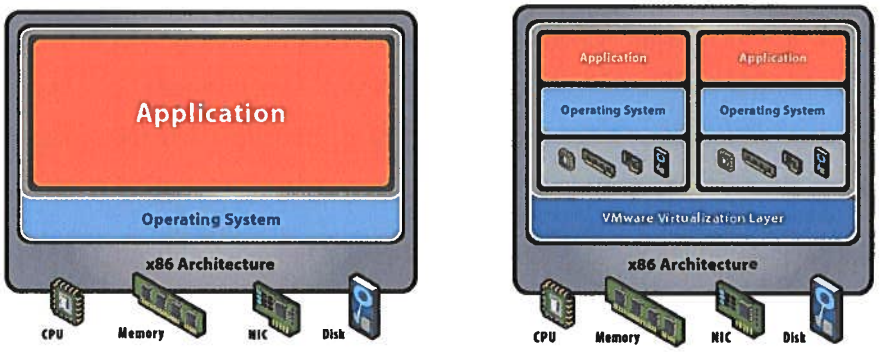

Figura 1 – Esquema representativo do processo de virtualização (VMware, 2006).

A implementação de uma infraestrutura de virtualização é considerada como não disruptiva, uma vez que a experiência do utilizador não é afetada. No entanto, permite aos administradores a vantagem de poderem gerir diversos recursos da empresa, o que permite aos gestores das TI ter uma resposta mais dinâmica face às necessidades organizacionais e fazer uma melhor gestão dos investimentos na infraestrutura (VMware, 2006) .

Com a memória virtual criada pelo processo de virtualização, por exemplo, o desenvolvimento de *software* tem acesso a maior capacidade de memória do que aquela que está realmente instalada, através da troca de dados para o armazenamento em disco (Hagen, 2008). De igual forma, técnicas de virtualização podem ser aplicadas a outras camadas de infraestruturas de TI, incluindo redes, armazenamento, *hardware* de computadores portáteis ou servidores, sistemas operativos e aplicações (VMware, 2006).

Em sistemas não virtualizados, apenas é permitida a existência de um único sistema operativo por cada servidor e um *software* e *hardware* fortemente indissociáveis, o que impossibilita a execução de várias aplicações no mesmo servidor e implica um desaproveitamento de recursos. Num sistema virtual é permitida uma independência entre o *hardware* dos sistemas operativos e as aplicações, sendo que qualquer máquina virtual pode ser aprovisionada independentemente do sistema. Na gestão dos sistemas operativos e das aplicações estes são tratados com uma única unidade encapsuladas numa máquina virtual (VMware, 2006).

O sistema físico virtual é responsável por executar o *software* que permite implementar a virtualização, sendo este denominado com o *Virtual Machine Monitor* (VMM) ou *Hypervisor* (Galvin, 2009). Ou seja, VMM é uma camada de *software* que virtualiza todos os recursos físicos de uma máquina permitindo a definição e suporte da execução de várias máquinas virtuais.

Existem dois tipos de virtualização, o Tipo I e o Tipo II (Figura 2) (VMware, 2006; Padhy, Patra, & Satapathy, 2011; Rodríguez-Haroa, et al., 2012). O Tipo I, também denominado por *Bare-Metal* define-se por ser executado diretamente num *hardware* físico, não dependendo de qualquer sistema operativo. Este assume a função de alocar os recursos de sistema das várias máquinas virtuais. São exemplos deste tipo de *Hypervisor* o VMware ESX (*Enterprise*) e Xen (VMware, 2006; Padhy, Patra, & Satapathy, 2011). No caso do Tipo II a VMM é executada como uma aplicação em um Sistema Operativo, sendo que este controla o estado do *hardware* denominado por sistema operativo *Guest*. Exemplo deste tipo de virtualização é o VMware GSX (*workstation*) e UML (*User-Mode Linux*). Este tipo de virtualização apresenta-se como sendo mais vulnerável do que o tipo I (Padhy, Patra, & Satapathy, 2011).

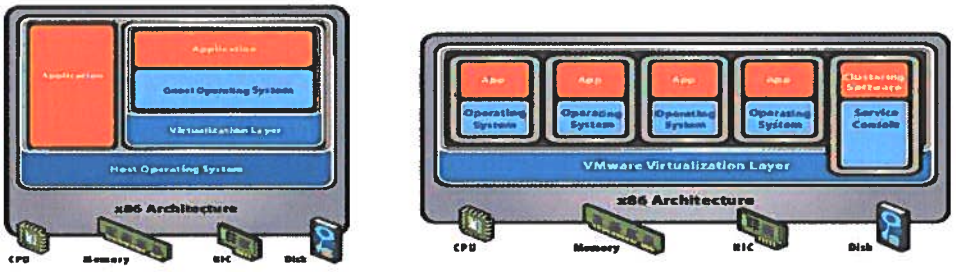

Figura 2 – Diagramas representativos das diferentes estruturas associadas ao tipo de virtualização I (esquerda) e II (direita) (VMware, 2006).

2.4. Benefícios e Riscos

As vantagens do uso da virtualização são inúmeras e em várias vertentes:

Fiabilidade e desempenho: qualquer falha que exista em termos de *software* numa máquina virtual não irá afetar as outras máquinas virtuais já que estas contêm ferramentas de monitorização e depuração para que não haja perda de produtividade e contenção de erros (Menascé, 2005; Uddin & A., 2011);

- **Custo:** é possível obter uma redução de custos consolidando servidores de menores dimensões em outros mais poderosos. A maior redução de custos é originada pelo facto de a virtualização reduzir a necessidade de recurso a *hardware*, mão-de-obra, espaço físico e licenças de *software*. Existem mesmo citações de redução de custos rondando entre os 29 a 64% (VMware, 2002; Menascé, 2005);
- **Adaptabilidade:** Alterações na intensidade dos níveis de trabalho podem ser facilmente realizadas através da troca de recursos e prioridades entre máquinas virtuais (Menascé, 2005) uma vez que existe compatibilidade binária através de todo o *hardware* e *software* o que melhora a capacidade de gerência entre os diferentes componentes do processo de virtualização (Uddin & A., 2011);
- **Partilha de recursos:** uma vez que o *software* de uma máquina virtual está completamente inserido na VMM, é relativamente fácil migrar as máquinas virtuais para outras plataformas de modo a melhorar a performance através do balanceamento da carga de trabalho afeto a cada parte do *software* (Uhlig, et al., 2005; Menascé, 2005);
- **Aplicações herdadas:** mesmo que uma dada empresa decida mudar para um novo sistema operativo é possível continuar a executar as tarefas herdadas no antigo sistema operativo que funciona como convidado na máquina virtual, o que reduz fortemente os custos de migração de sistemas (Menascé, 2005).
- **Segurança:** quando uma máquina virtual se encontra comprometida é fácil de desligar e reiniciar todo o sistema com uma cópia não corrupta da máquina virtual, cópia essa que pode ser obtida com frequência e de forma rápida (Leja, et al., 2008).

Apesar de as máquinas virtuais serem bastante fiáveis e apresentarem inúmeros benefícios não deve ser descurado o facto da possibilidade de esta falhar ou diminuir fortemente a performance já que tem múltiplas aplicações a funcionarem ao mesmo tempo (Uddin & A., 2011; Leja, et al., 2008). É necessário ultrapassar os desafios de centralizar os dados numa infraestrutura física, com alta densidade dinâmica, sistemas de alimentação e refrigeração.

Outro dos riscos que se enfrenta com a virtualização é a elevada proliferação de servidores, associada ao facilitismo e rapidez com que as máquinas virtuais podem ser geradas (Leja, et al., 2008). Assim, é necessário conter ou até mesmo reverter a expansão dos servidores em excesso, de modo a que não haja no futuro problemas de gestão e alavancagem de ativos físicos.

Apesar de a virtualização estar associada a uma generalizada redução de custos, a demanda por mão-de-obra especializada para a gestão e implementação dos sistemas virtuais pode apresentar-se como um acréscimo neste aspeto, já que especialistas demasiado específicos podem ser raros e isso requerer elevados pagamentos em termos salariais (Leja, et al., 2008).

3. Hyper-V

3.1. Definição e arquitetura

Atualmente, o Hyper-V apresenta-se como a solução para a virtualização de um sistema mais utilizada (Apolinário, 2015). A plataforma Hyper-V pertence à Microsoft e tem sofrido inúmeros desenvolvimentos e aperfeiçoamentos desde a sua primeira apresentação em 2008, como componente do sistema operativo Windows Server 2008 (Microsoft, 2013). Para as empresas, o Hyper-V apresenta-se como uma ferramenta que consolida múltiplos servidores em máquinas virtuais separadas que são executadas num único servidor físico. Permite, ainda, executar diferentes sistemas operativos (Linux incluído), num único servidor físico tirando o maior partido de um sistema de 64-bit. (Microsoft, 2013)

Comparativamente com os seus competidores, a Microsoft entrou no mercado um pouco tardiamente através da aquisição da empresa Connectix, em 2003, desenvolvendo o Virtual PC, uma aplicação do ambiente de trabalho que os utilizadores podiam instalar na base do sistema operativo (Shah, 2013). O lançamento deste produto permitiu, no âmbito do mercado individual e pessoal, investir no desenvolvimento do Microsoft Virtual Server e em várias versões deste, nomeadamente o Virtual Server 2005 e o Virtual Server 2005 R2 (Morimoto & Guillet, 2008). Os desenvolvimentos subsequentes ao lançamento através dos Windows Server 2008 R2 em 2009 e Windows server 2012 permitiram o aparecimento de inúmeras funcionalidades inexistentes na primeira versão, tais como a Memória Dinâmica e o Remote FX (Microsoft, 2013).

O diagrama representado no esquema da Figura 3 pretende representar as características principais da arquitetura do Hyper-V. A estrutura típica do Hyper-V segue o modelo de virtualização do tipo I, ou seja, *Bare-Metal*, que é executado num *hardware*. O *hypervisor* divide-se em partição parental ou original (*parental partition*) e uma ou mais partições secundárias (*child partition*). Uma partição é uma unidade isolada para a qual é alocado espaço para memória física e processadores virtuais (Shah, 2013).

A partição original, a primeira a ser criada, irá conter o acesso a aparelhos de *hardware* e ao conjunto local de virtualização. Esta partição pode gerar outras partições

secundárias e gerir todos os componentes a elas associados. Nestas partições secundárias podem ser executados todos os componentes da máquina virtual convidada.

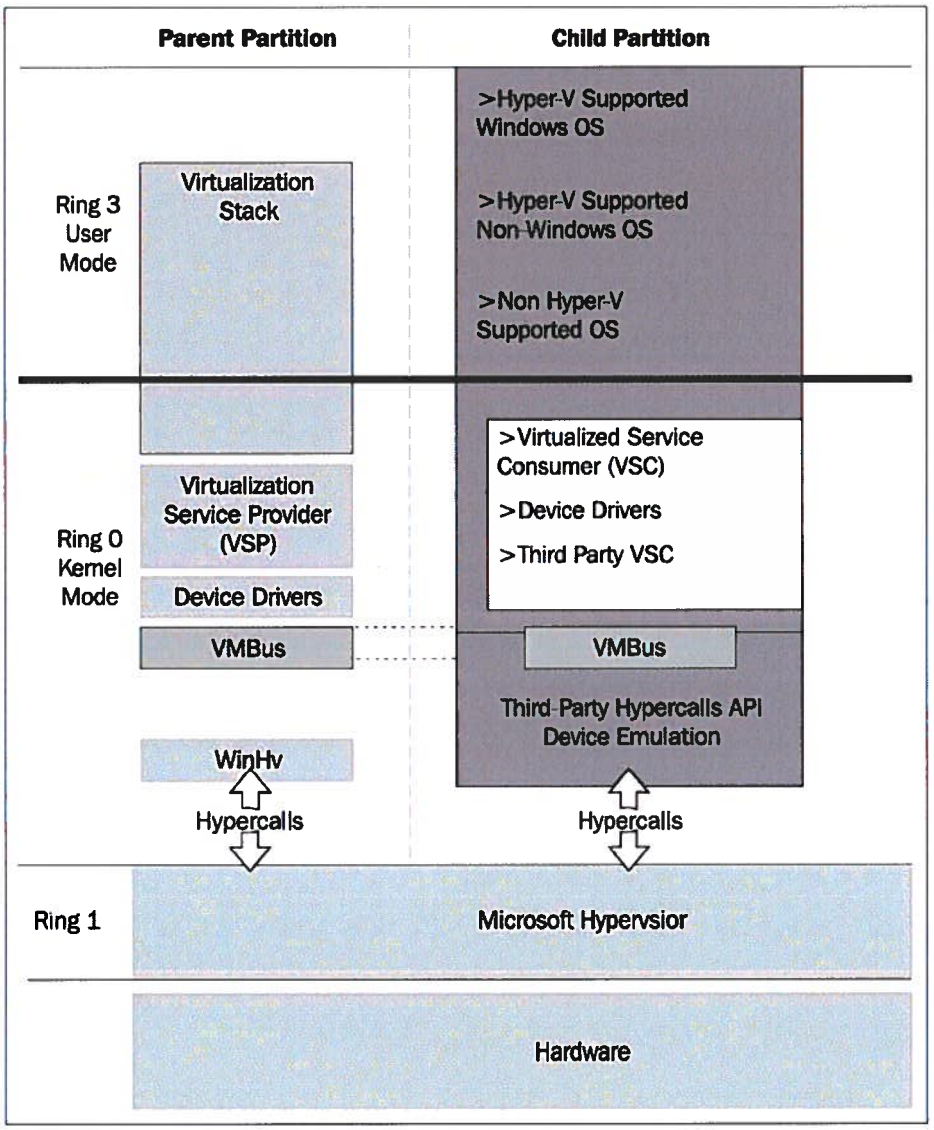

Figura 3 – Diagrama representativo da arquitetura do Hyper-V (Imagem retirada de (Shah, 2013)).

3.2. Vantagens do Hyper-V

Apesar das inúmeras melhorias desde a primeira versão do Hyper-V da Microsoft (Figura 4), existe uma vantagem que tem sido transversal desde o início, que é o facto de esta ferramenta de virtualização ser distribuída gratuitamente, ficando acessível para vários mercados (Apolinário, 2015).

O Uso do Hyper-V estende-se a múltiplos sistemas operativos, num único servidor e a última versão (Windows Server 2012 R2 Hyper-V) permite uma maior escalabilidade, novos mecanismos de fiabilidade, novas características de armazenamento e interligação, e maior integração com o *hardware*.

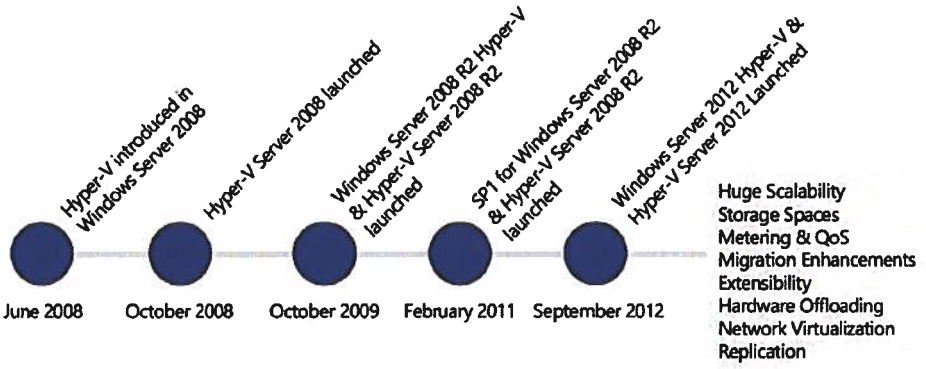

Figura 4 – Linha temporal do lançamento de servidores Hyper-V antes da versão Windows Server 2012 R2. (Imagem retirada de (Microsoft, 2013).

Os quatro pontos chave do Hyper-V que se apresentam como vantajosos em relação a outras plataformas de virtualização, nomeadamente a VMware (Microsoft, 2013), são:

- **Escalabilidade, performance e densidade:** o Hyper-V surge como uma plataforma que permite a gestão de sistemas de elevadas performances e necessidades a um baixo custo. Os pontos de integração entre o *hardware* e o Hyper-V permitem grandes níveis de trabalho, o que permite às empresas tirarem partido de sistemas físicos de maiores dimensões;

- **Segurança e multiplicidade:** o Hyper-V permite aos clientes, de forma segura e, em alguns casos, criptografada e facilmente controlável, aceder ao seu espaço de trabalho no ambiente virtual, permitindo o uso de infraestruturas virtualizadas como um serviço externo à empresa;
- **Flexibilidade:** Numa base de dados moderna, os clientes exigem agilidade de modo a responder de forma rápida e eficiente às exigências na área dos negócios. O Hyper-V, através das inovações *Live Migration* e *Network Virtualization*, consegue fornecer a capacidade de mover trabalho de forma flexível em torno da infraestrutura. Para além disso, para clientes que requerem Windows e Linux, o Hyper-V apresenta-se como uma boa plataforma alternativa;
- **Disponibilidade e resiliência:** o Hyper-V permite, em caso de perda de dados ou de paragem de funcionamento, restaurar rapidamente o sistema até mesmo em outra localização geográfica. Está garantido que o cliente consegue ter disponível de forma contínua todas as suas aplicações mais críticas.

4. ADO.NET

4.1. História e definição de ADO.NET

No início do desenvolvimento das formas de acesso a bases de dados era necessário que um programador tivesse um vasto conhecimento da Interface de Programação de Aplicativos de modo a implementar o seu código para cada versão do *driver*, o que consumia muito tempo. No entanto, com a demanda pelo crescimento do mercado, a Microsoft, em 1990, criou o padrão ODBC (*Open DataBase Connectivity*) que criava uma interface e padrão único para o programador (Lendvai & Shi, 2007; Patrick, 2010; Húngaro, 2016).

O grande avanço que surgiu pela necessidade de evolução, deu-se com o desenvolvimento do OLEDB, um tipo de Interface de Programação de Aplicações, para o qual se criou o ADO (*ActiveX Data Objects*) com o objetivo de consumir os recursos oferecidos pelo OLEDB. Desta forma, foi permitido criar várias camadas de acesso a bancos de dados, simplificando e padronizando o seu acesso.

A ADO.NET, introduzida em 2000 pela Microsoft, é uma tecnologia amigável usada para aceder a bancos de dados da plataforma .NET que se encontra integrada ao .NET *Framework*, um ambiente de desenvolvimento de aplicações. Esta oferece diversas classes que permitem realizar um elevado número de tarefas associadas ao acesso e manutenção de dados que podem ocorrer sem qualquer intermediário (Lendvai & Shi, 2007; Patrick, 2010); consegue fazer a gestão tanto de dados internos – dados criados na memória e usados exclusivamente na aplicação – e dados externos – dados armazenados numa zona aparte da aplicação, tal como num ficheiro de texto (Patrick, 2010). Independentemente da fonte, a ADO.NET generaliza os dados relevantes e apresenta-os sob a forma de tabelas compostas por linhas e colunas.

4.2. Propriedades da ADO.NET

No desenvolvimento da ADO.NET, a Microsoft dotou esta aplicação com características especiais para facilitar o desenvolvimento no contexto do acesso padronizado a bases de dados (Húngaro, 2016; Patrick, 2010):

- **Acesso desconectado:** para manipular os dados não é necessário manter ativa uma conexão com a fonte dos dados. Isto é muito importante em aplicações onde o bom desempenho é fundamental, como páginas *Web* e aplicações para dispositivos móveis. Este ponto terá sido o principal foco da equipa de desenvolvimento da ADO.NET e a principal mudança em relação ao ADO tradicional;
- **Integração com o XML:** o XML é um padrão aberto para transferência de dados. A ADO.NET faz uma ligação nativa em XML, facilitando a interoperabilidade entre plataformas heterogéneas;
- **Tudo pode ser uma fonte:** diferente da ADO, que era focada no banco de dados, a ADO.NET tem a capacidade de conseguir extrair dados de diversas fontes, tais como: arquivos XML e qualquer fonte que puder ser acedida via XML ou OLEDB;
- **Representação comum dos dados:** permite que os dados sejam manipulados pelos mesmos objetos, independente da sua fonte ser um banco de dados, um arquivo XML ou qualquer outra. Isso permite que os programadores não se preocupem em como manipular dados de diversas fontes, pois uma vez conhecendo a arquitetura da ADO.NET, a manipulação desses dados é feita sempre de forma muito semelhante;
- **Armazenamento:** é possível armazenar dados em *cache* para melhorar a performance de aplicações ASP.NET.

4.3. Principais componentes da ADO.NET

A ADO.NET herda os benefícios da *Framework* .NET, baseada numa visão orientado ao objeto e acoplada pelo modelo *self-describing* (Lendvai & Shi, 2007).

Sendo o ADO.NET uma extensão/evolução do modelo ADO em que este providencia uma abordagem multifacetada no que respeita à gestão dos dados, este modelo apresenta-se como uma mudança no paradigma, em que o conjunto de objetos são otimizados para tarefas específicas. O acesso de dados ADO.NET baseia-se nas classes *Data Provider* e *DataSet* (Figura 5). A classe *Data Provider* pertence ao ambiente de programação conectado, enquanto que a classe *Dataset* pertence ao ambiente desconectado (Hamilton & MacDonald, 2003).

4.4. Dados Conectados e Desconectados

O ambiente conectado (*Data Provider*) promove o acesso à base de dados apenas por reencaminhamento ou leitura e permite executar comandos para a fonte dos dados. Assim, esta classe providencia uma forma comum de trabalhar com dados conectados, independentemente da fonte de dados subjacente. Este ambiente inclui as classes *Connection*, *Command*, *DataReader*, *Transaction*, *ParameterCollection* e *Parameter* (Hamilton & MacDonald, 2003).

Connection:

Mantém as informações necessárias para fazer a ligação à fonte de dados através de uma sequência. A cadeia de ligação contém informações como o nome da fonte de dados e sua localização, credenciais e configurações. A classe *Connection* tem métodos para abrir e fechar a conexão, para transações a serem iniciadas na conexão, assim como o controlo de outras propriedades da conexão.

Command:

Executa instruções de SQL ou procedimentos armazenados na fonte de dados. Esta classe tem o objeto *ParameterCollection* que contém objetos *Parameter* que permitem instruções SQL com parâmetros e procedimentos armazenados para serem usados na fonte de dados.

DataReader:

Fornece o acesso à base de dados apenas por reencaminhamento ou leitura. Está otimizado para velocidade. O *DataReader* é instanciado através do objeto *Command.*

Parameter:

Permite que os parâmetros para ambas as consultas com parâmetros e procedimentos armazenados sejam definidos e ajustados para valores apropriados. É dado acesso à classe *Parameter* através do objeto *ParametersCollection* dentro de um objeto *Command.* Suporta parâmetros de entrada e saída, bem como valores de retorno a partir de procedimentos armazenados.

Transaction:

Permite que as transações sejam criadas de modo a que várias alterações nos dados de uma fonte de dados sejam tratadas como uma única unidade de trabalho.

DataAdapter:

Faz a ligação entre a fonte de dados e o *DataSet* ou *DataTable*. O *DataAdapter* envolve as classes conectadas para permitir esta funcionalidade. Fornece um método para recuperar dados de um objeto desconectado e um método para reconciliar os dados modificados no objeto desconectado com a fonte de dados. A classe *CommandBuilder* pode gerar a lógica de conciliar mudanças em situações simples; uma lógica personalizada pode ser fornecida para lidar com situações complexas e otimizar o desempenho.

O ambiente desconectado permite que os dados recuperados da fonte de dados possam ser manipulados *offline* e depois reconciliados com a fonte de dados. As classes desconectadas fornecem um modo comum para funcionar com dados desconectados de forma independente da fonte de dados subjacente. Este inclui as classes *DataSet*, *DataTable*, *DataColumn*, *DataRow*, *Constraint*, *DataRelation* e *DataView* (Hamilton & MacDonald, 2003).

DataSet:

Fornece uma maneira consistente para lidar com dados desconectados de forma completamente independentemente da fonte de dados. O *DataSet* é essencialmente uma base de dados relacional criada em memória que serve como um recipiente para os objetos *DataTable*, *DataColumn*, *DataRow*, *Constraint* e *DataRelation*. O formato XML serializa e transporta um *DataSet* que pode ser acedido e manipulado quer como XML ou através dos métodos e propriedades do *DataSet*, alternadamente; a classe *XmlDataDocument* representa e sincroniza os dados relacionais dentro de um objeto *DataSet* com o *XML Document Object Model (DOM XML)*.

DataTable:

Permite que os dados desconectados possam ser acedidos e modificados através de um conjunto de classes *DataColumn* e *DataRow*. O *DataTable* permite que restrições, como chaves estrangeiras e restrições exclusivas, possam ser definidas usando a classe *Constraint*.

DataColumn:

Corresponde a uma coluna numa tabela. Esta classe armazena metadados sobre a estrutura da coluna que, juntamente com a classe *Constraint*, definem o esquema da tabela. O *DataColumn* também pode criar colunas de expressão com base em outras colunas na tabela.

DataRow:

Corresponde a uma linha na tabela e pode consultar e atualizar dados na classe *DataTable*. O *DataTable* expõe os objetos *DataRow* através do objeto *DataRowCollection* que ele contém. O *DataRow* armazena as alterações feitas nos dados contidos nas suas colunas, armazenando valores tanto originais como atuais. Isto permite que as alterações possam ser posteriormente canceladas ou reconciliadas com a fonte de dados.

Constraint:

Permite a colocação de limitações nos dados armazenados dentro de uma *DataTable*. Para manter a integridade dos dados podem ser criadas restrições únicas ou com chaves estrangeiras.

DataRelation:

Fornece uma forma para indicar uma relação entre diferentes objetos *DataTable* dentro de um *DataSet*. O *DataRelation* relaciona colunas nas tabelas pai e filha que permitem a navegação entre estas tabelas e a integridade referencial, executada por meio de atualizações em cascata e eliminações.

DataView:

Permite que os dados, uma vez recuperados de um *DataSet* ou *DataTable*, possam ser vistos de formas distintas. Permite que os dados sejam classificados com base em valores das colunas e por um subconjunto dos dados filtrados, de modo a que apenas as linhas correspondentes a critérios específicos sejam exibidas.

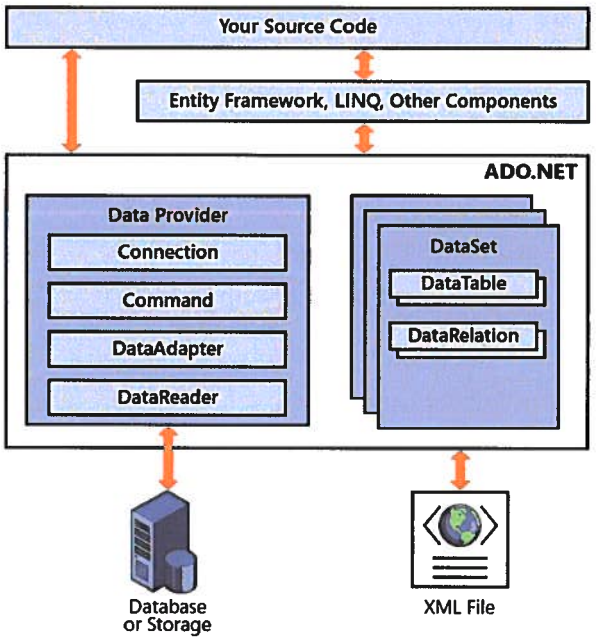

Figura 5 – Principais componentes de ADO.NET (Imagem retirada de (Patrick, 2010)).

Como verificado, a componente .NET providencia meios para comunicar com qualquer origem de dados. Por exemplo, cada fonte de dados dispõem um conjunto de classes que variam consoante o motor de Base dados (SGBD) como é o caso do SQL, Oracle e outro. Segundo os autores (Lendvai & Shi, 2007) o modelo ADO.NET introduz um nível de abstração onde existe uma separação entre os dados e os processos e a forma como são recuperados e atualizados, como demonstrado na Figura 6.

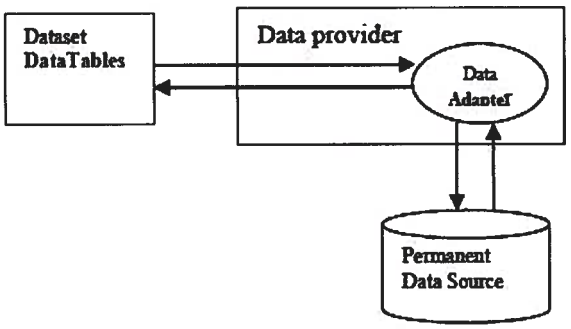

Figura 6 – Abstração no modelo ADO.NET. (Imagem retirada de (Lendvai & Shi, 2007)).

4.5. Situações para o uso de ADO.NET

A utilização do modelo ADO.NET tem benefícios quando é usado nas seguintes circunstâncias (Lendvai & Shi, 2007):

- Quando existe necessidade para uma gestão e controlo de *updates* em grupo;
- Quando a usabilidade tem prioridade em *updates* de dados desconectados com dados de origem;
- Quando é requerida escalabilidade;
- Quando a utilização de procedimentos armazenados em *updates* em grupo é importante.

5. SQL Server 2014

5.1. Sistemas de Gestão de Base de Dados (SGBD)

O objetivo de um Sistema de Gestão de Base de Dados (SGBD) é fornecer um ambiente que seja conveniente e eficiente para armazenamento e recuperação de informações (Hellerstein, Stonebraker, & Hamilton, 2007). Para que um sistema SGBD funcione é necessário que possa garantir a segurança em caso de falhas de um sistema ou em tentativas de acesso não autorizado, permitir cópias de segurança e recuperação, controlar a concorrência e executar os comandos de seleção, inserção, atualização ou apagar. Um SGBD é constituído por uma coleção de dados inter-relacionados e por um conjunto de programas desenvolvidos que permitem o acesso e a manipulação dos dados. A gestão destes dados implica a definição não só das estruturas em que irão ser armazenados, mas também os mecanismos para poder manipular esses dados.

Na atualidade, e para fazer frente à crescente necessidade da gestão de dados não estruturados, como os encontrados na maioria dos documentos e páginas de internet, existem vários SGBD, nomeadamente o FireBird, Oracle, MySQL, PostgreSQL e Microsoft SQL Server (do tipo relacional), o qual irá ser explorado mais em detalhe.

5.2. Evolução dos servidores SQL

O primeiro SQL Server foi lançado pela Microsoft em 1988. Esta primeira plataforma consistia numa versão especial do Sybase.

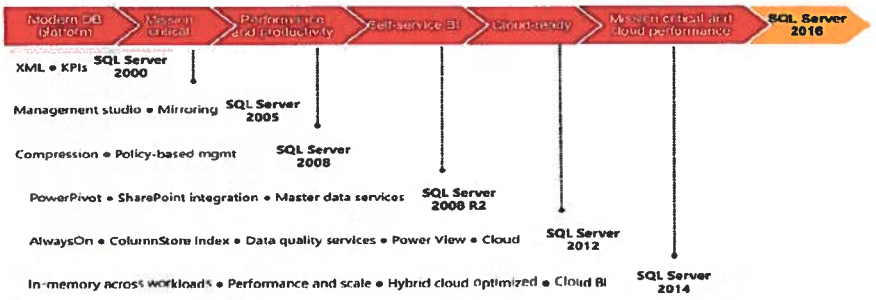

Figura 7 – Funcionalidades adicionadas ao SQL Server durante a sua evolução (Imagem retirada de (Microsoft, 2014)).

Após o primeiro lançamento várias melhorias e inovações foram levadas a cabo nos últimos 15 anos, o que se encontra esquematizado na Figura 7, tendo havido um grande investimento por parte da Microsoft nesta área. Cada edição destina-se não só a públicos-alvo diferentes, mas também tem em conta as diferentes cargas de trabalho. Apesar de a última versão ser o SQL Server 2016, o foco será na versão de 2014.

Conforme o *White Paper* publicado pela Microsoft (Microsoft, 2014), com a versão do SQL Server 2014 foram implementadas uma série de novas funcionalidades que permitiram melhorar o produto em si, quando em comparação com as versões anteriores. Segundo o mesmo artigo foram feitas melhorias no que diz respeito à gestão de memória resultando, em alguns casos, numa melhoria de performance até 10x em processamento de transações. Com esta versão ficaram, ainda, disponíveis melhorias nas soluções de alta disponibilidade e recuperação em caso de desastre, o que permite disponibilizar redundância através de métodos implementados em vários centros de dados e implementar igualmente métodos que garantem maior disponibilidade e proteção de dados para aplicações com criticidade elevada. Redundância em ambientes em nuvem (*Cloud*), nomeadamente o Microsoft Azure, permitindo que os clientes possam ter uma redução do custo de investimento no que diz respeito a servidores de *standby* é também uma das características transversais a esta versão de 2014.

A Figura 8 demonstra como as várias atualizações da plataforma SQL 2014 a tornaram menos vulnerável e confiável, como demonstrou o estudo realizado pelo *National Institute of Standards and Technology* (Microsoft, 2014). Esta performance foi conseguida através da separação do papel do administrador na base de dados do papel do administrador de sistemas.

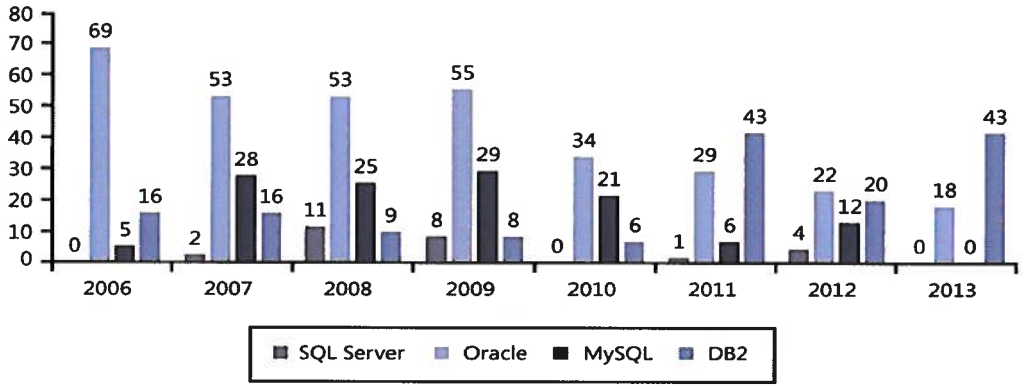

Figura 8 – Medição da vulnerabilidade de vários sistemas desde 2006 até 2013, tal como publicado no relatório National Institute of Standards and Technology. (Imagem retirada de (Microsoft, 2014)).

A integração de novos métodos de Segurança, *Data Warehousing* e *Advanced Compression*, permitiram otimizações que reduziram a ocupação de espaço de armazenamento até 87% e uma redução de ocupação no que diz respeito a dados de *Backup*. Por último, foram desenvolvidas novas funcionalidades e implementação de ferramentas de *Business Intelligence*.

5.3. Principais componentes do Microsoft SQL Server

Os principais componentes do Microsoft SQL Server são:

- ***Database Engine***: Fornece acesso de forma controlada e o processamento de transações para atender aos requisitos dos aplicativos. Permite armazenar, recuperar, processar e proteger dados;

- ***SSRS (SQL Server Reporting Services):*** Esta componente fornece ferramentas e serviços prontos para criar, implementar e gerir relatórios. Possui recursos de programação para que possa ser possível estender as funcionalidades e personalizar os relatórios;

- ***SSAS (SQL Server Analysis Services):*** Os seus dois componentes são o OLAP (*On-Line Analytical Processing*) e o *Data Mining*. Fornece a solução de *Business intelligence* e é responsável por transformar uma grande quantidade de dados em conjuntos essenciais de informação;
- ***SSIS (SQL Server Integration Services):*** Responsável por extrair e transformar os dados de diversas fontes como arquivos de dados XML, arquivos simples e bases de dados relacionais e transfere para um ou mais destinos.

5.4. Funcionamento do SQL 2014

Segundo o autor Butterfield a linguagem SQL, acrónimo para *Sequential Query Language,* é utilizada em combinação com o SQL Server, para armazenar e aceder aos dados sobre redes de elevada densidade, de uma forma rápida, eficiente e segura (Butterfield, 2007). Uma base de dados SQL é relacional, em que comandos são utilizados para criar instantâneos de dados dentro das tabelas de base de dados SQL utilizando consultas (*queries*). Essas consultas são pedidos realizados por um utilizador através das tabelas de bases de dados SQL, em que o computador realiza uma procura pela informação pedida. Podem ser utilizados vários termos ou comandos como SELECT, INSERT ou DELETE para aceder aos dados nas tabelas.

Um exemplo de uma consulta nestas bases de dados poderá ser, tal como descrito em (Butterfield, 2007):

> *"For example, if I had a table that consisted of one thousand people, separated by their first and last names, I could query the table to find all of the people with the first name of "Bob" by using the SELECT*

> *command. Conversely, I could delete all instances of the name "Bob" by using the DELETE command. I could also insert a new "Bob Smith" by using the INSERT command"*

Segundo o mesmo autor, o SQL é uma ferramenta de linguagem de Base de Dados muito popular, porque é bastante mais eficiente para usar em redes onde existe uma maior quantidade de dados, do que em bases de dados to tipo *standalone*, como o Microsoft Access (Butterfield, 2007).

Segundo a Microsoft, os motores/base de dados providenciam um armazenamento do tipo relacional para aplicações cliente-servidor (Microsoft, 2014). À medida que as necessidades das empresas vão surgindo, cada vez mais capacidades vão sendo adicionadas às aplicações, tais como segurança, alta disponibilidade ou *business intelligence*. A forma como essas necessidades são respondidas define a forma como o cliente final suportará o encargo da implementação das aplicações usando a respetiva base de dados (Microsoft, 2014).

A **Error! Reference source not found.** reflete o sumário de todas as funcionalidades presentes no SQL Server 2014.

Tabela 1 – Funcionalidades do SQL Server 2014.

Características	Detalhes
Memória	*In-Memory OLTP, In-Memory Columnstore, Buffer Pool Extension*
Disponibilidade e Recuperação	*AlwaysOn*
Segurança avançada	*Transparent Data Encryption, Extensible Key Management, SQL Server Audit, User-Defined Server Roles, Default Schema for Groups, Contained Database Authentication, Backup Encryption*
Data Warehousing	*Change Data Capture, Table Partitioning, Integration Services*
Compressão avançada	*Storage and Backup Compression*
Gestão	*Resource Governor, Distributed Replay, Control Point, Database Tuning Advisor, Performance Data Collector, Policy Based Management*
Componente Não Relacional	*Spatial data support, FileTable*
Business Intelligence	*Analysis Services, Reporting Services, Data Mining, Semantic Model*
Gestão da informação da empresa	*Master Data Services, Data Quality Services*
Processamento de eventos	*StreamInsight*
Componentes em nuvem	*Backup database to Azure, Data files in Azure, AlwaysOn Disaster Recovery in Azure, Deploy database to Azure tool*

6. Projeto da Biblioteca Virtual

6.1. Âmbito

Este projeto consistiu no desenvolvimento de uma biblioteca virtual assente numa infraestrutura com servidores virtualizados. O objetivo desta biblioteca é permitir que utilizadores credenciados obtenham acesso a uma base de dados com diversos documentos publicados por autores, nomeadamente artigos científicos, livros ou revistas. Desta forma, pretende-se trazer valor acrescentado a todos os utilizadores de bibliotecas tradicionais, permitindo que, para além do acesso tradicional aos seus livros, o utilizador possa também requisitar uma máquina virtual com a qual terá acesso de forma rápida a diversas publicações em formato digital, por meio de uma pesquisa que pode ser feita pelo nome do livro, nome do autor, editora, entre outros.

O utilizador que pretenda requisitar uma máquina virtual deve dirigir-se ao Bibliotecário e efetuar o seu pedido. Em seguida, será preenchido um formulário e ser-lhe-ão atribuídas as suas credenciais de acesso remoto a uma máquina virtual disponibilizada automaticamente através da utilização do SCVMM (*System Center Virtual Machine Manager*). O utilizador poderá aceder remotamente a esta máquina virtual, quer através do seu computador pessoal, quer através de um dos computadores que a biblioteca poderá disponibilizar para o efeito.

Ao efetuar *login* na máquina virtual, o utilizador terá acesso a uma aplicação *windows* muito intuitiva em que, para encontrar o livro que pretende, apenas terá de introduzir os termos da sua pesquisa e serão apresentadas imediatamente todas as publicações que contenham o termo pesquisado. Os livros pesquisados e guardados no computador remoto poderão ser transferidos para uma área de armazenamento de documentos, disponibilizada na *Cloud* da Microsoft.

Ao Bibliotecário e às equipas responsáveis pela gestão e manutenção da plataforma, é disponibilizada uma máquina virtual, com a qual têm acesso à área de administração para fazer a gestão dos conteúdos da biblioteca virtual, através da adição

ou atualização dos diversos documentos. Têm ainda acesso a uma OU (*Organizational unit*) no *Active directory*, onde podem fazer a gestão dos utilizadores da biblioteca virtual.

6.2. Laboratório Virtual

Para facilitar a apresentação deste projeto, e por questões de portabilidade, optou-se pela utilização da tecnologia *hypervisor* disponibilizada pelo VMware Workstations 12, que permite colocar todas as máquinas virtuais em armazenamento externo, podendo assim ser transportado para qualquer outro computador e aí ser iniciado o laboratório.

6.3. Criação das máquinas virtuais em laboratório

Após a instalação do VMware Workstation 12 (Figura 9), iniciou-se a criação das três máquinas virtuais essenciais para colocar todo o ambiente em funcionamento.

Na consola do VMware Workstation criaram-se as máquinas virtuais com as seguintes características apresentadas na Tabela 2.

Tabela 2 – Características dos servidores utilizados para a criação das máquinas virtuais.

Servidor	Roles e Software	Sistema operativo	CPU Cores	Memória	Disco
DC1	Active directory domain services, DNS, DHCP	Windows server 2012 R2 Standard	2	1GB	100GB
HV1	Hyper-V, System Center Virtual Machine Manager 2012	Windows server 2012 R2 Datacenter	8	6GB	200GB
SQL	Microsoft SQL server 2014	Windows server 2012 R2 Datacenter	8	2GB	100GB

A criação das máquinas virtuais foi efetuada seguindo os passos que são disponibilizados pelo "*Wizard*" da consola VMware.

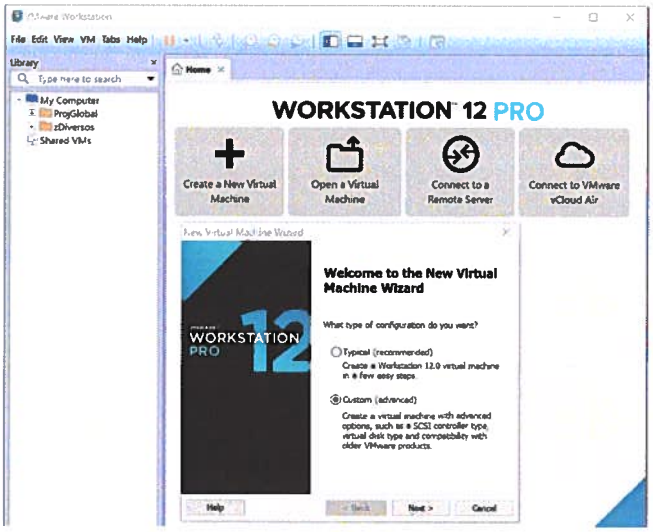

Figura 9 – Criação de máquinas virtuais, opções avançadas.

Após a escolha da compatibilidade de *hardware* da máquina virtual para a versão mais atualizada, selecionou-se a imagem ISO[1] do CD[2] que contém a instalação do Windows Server 2012 R2 Standard, que é a versão que irá ser instalar no servidor DC1 (Figura 10). A versão do sistema operativo a ser instalada nos restantes servidores encontra-se especificada na **Error! Reference source not found.**.

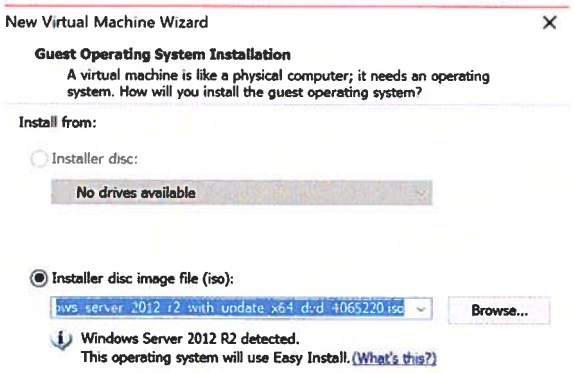

Figura 10 – Imagem ISO da instalação do sistema operativo.

[1] Imagem de um disco ótico, seja um CD (*Compact Disc*), DVD (*Digital Video Disc*), ou Blue-Ray.
[2] Disco ótico

Prosseguiu-se com a identificação do nome das máquinas virtuais e da sua localização no disco físico, onde ficarão guardados os discos virtuais (Figura 11).

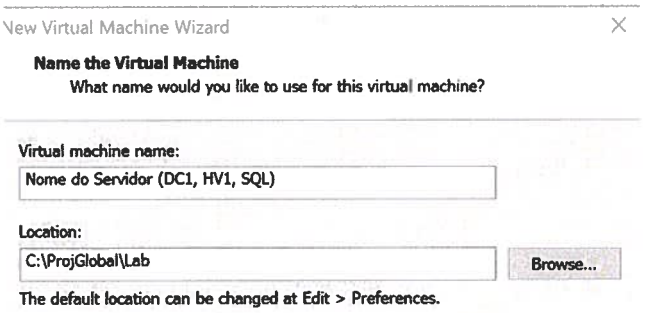

Figura 11 – Nome e localização das máquinas virtuais.

O tipo de *firmware* utilizado para o arranque da máquina virtual foi a BIOS[3]. Para o presente projeto a utilização de EFI[4] pode ser dispensada, uma vez que apesar deste apresentar inúmeras vantagens, neste caso a sua utilização não é justificada.

O número de processadores virtuais, assim como a quantidade de memória virtual a utilizar em cada uma das máquinas virtuais, encontra-se especificada na **Error! Reference source not found.** e Figura 12 e 13. Estes valores poderão ser ajustados conforme a quantidade de recursos que sejam disponibilizados pelo *hardware* da máquina onde se encontra instalado o *Hypervisor*.

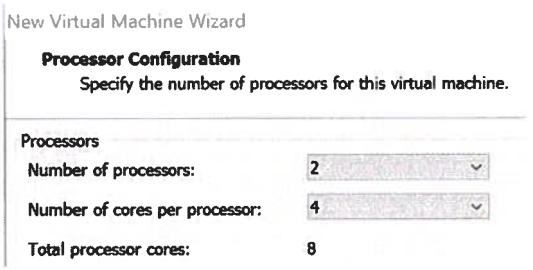

Figura 12 – Número de processadores virtuais.

[3] *Basic Input/Output System*
[4] *Extensible Firmware Interface*

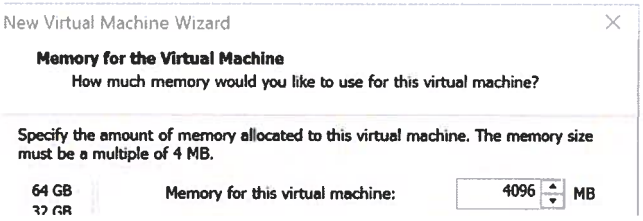

Figura 13 – Quantidade de memória virtual.

Para cada máquina virtual optou-se por adicionar dois adaptadores de rede:

- NAT[5]: permite que as máquinas virtuais utilizem o mesmo acesso de internet que o sistema operativo onde se encontra instalado o *Hypervisor*. Com este acesso pode-se efetuar a atualização do sistema operativo das máquinas virtuais.
- "*host-only*": permite que todas as máquinas virtuais se encontrem na mesma rede que o seu *host*. Assim pode-se aceder remotamente às máquinas virtuais utilizando, por exemplo, o cliente *remote desktop* da Microsoft, a partir do *host*.

No seguimento da criação da máquina virtual, apenas se teve a possibilidade de adicionar um adaptador de rede, pelo que a opção foi o NAT (Figura 14) e, posteriormente, adicionou-se o *host-only*.

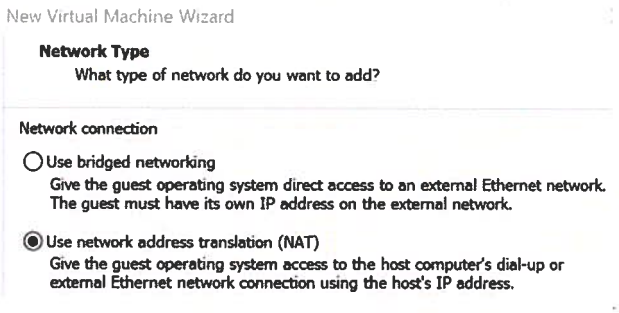

Figura 14 – Tipo de adaptador de rede.

[5] *Network Address Translation*

O tipo de controladora LSI Logic SAS é a recomendada e a única suportada para discos virtuais com sistema operativo Windows Server 2012. O tipo de disco virtual recomendado e selecionado foi o SCSI[6] pois, entre outras vantagens, é o que tem melhor desempenho.

O tamanho máximo atribuído ao disco de sistema de cada máquina virtual, encontra-se especificado na **Error! Reference source not found.** e Figura 15.

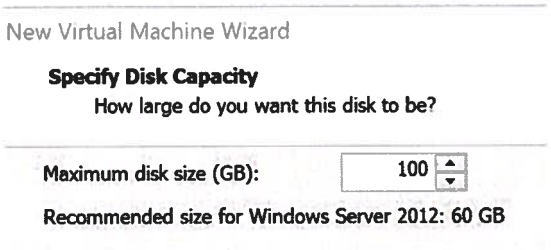

Figura 15 – Tamanho máximo do disco virtual.

Com os procedimentos já descritos, deu-se por finalizada a criação da máquina virtual. A Figura 16 mostra um resumo das configurações que foram efetuadas ao longo do processo de criação da máquina virtual.

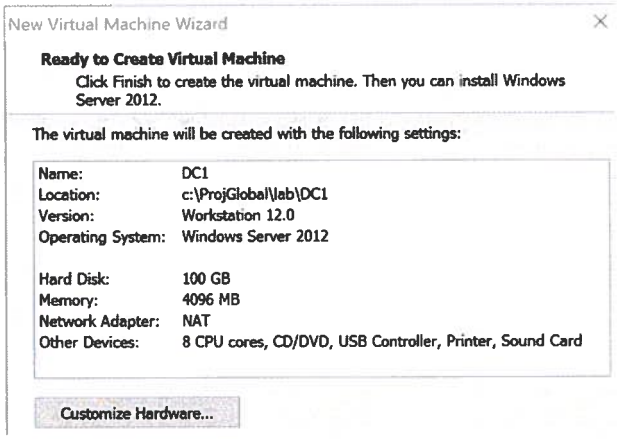

Figura 16 – Resumo das configurações da criação da máquina virtual.

[6] *Small Computer System Interface*

Ao finalizar a criação da máquina virtual é iniciada a instalação do sistema operativo, pelo que é necessário aguardar até que este finalize.

O processo acima descrito terá de ser efetuado para a criação das máquinas virtuais **DC1, HV1 e SQL.**

6.4. Configuração do sistema operativo

Depois de todas as três máquinas virtuais terem sido instaladas (**DC1, HV1 e SQL**) com o sistema operativo especificado na **Error! Reference source not found.**, é altura de se proceder à sua configuração e preparação para que este, para além das atualizações *Windows,* tenha também instalado todos os pré-requisitos necessários às especificidades de cada *role* e *Software* que aí será instalado.

A primeira etapa é desligar as máquinas virtuais e adicionar nas definições de cada uma um segundo adaptador de rede, que neste caso será do tipo "*host-only*".

De seguida volta-se a ligar as máquinas virtuais, altera-se o seu nome (

Figura 17), define-se o endereço IP fixo do adaptador de rede LAN (

Figura 18) e, de forma a ficarem de acordo com os nomes e IPs constantes na **Error! Reference source not found.**.

Tabela 3 – Configurações de rede dos servidores.

Nome	*IP*	*Subnet*	*Gateway*	*DNS*
DC1	10.19.36.2			
HV1	10.19.36.3	255.255.255.0	10.19.36.1	10.19.36.2
SQL	10.19.36.4			

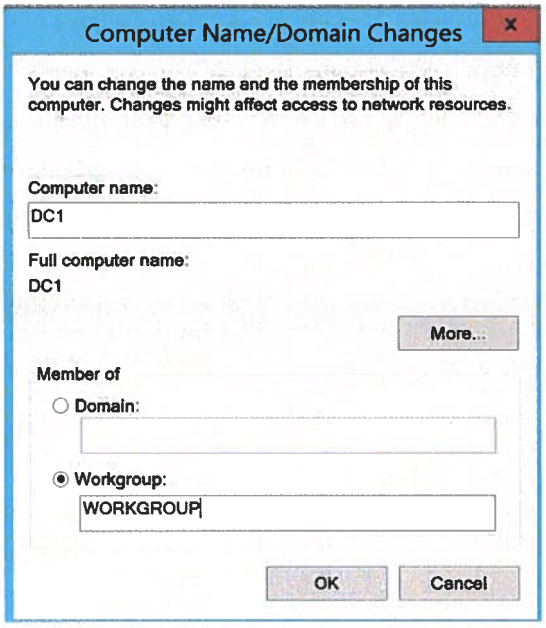

Figura 17 – Alteração de nome do servidor.

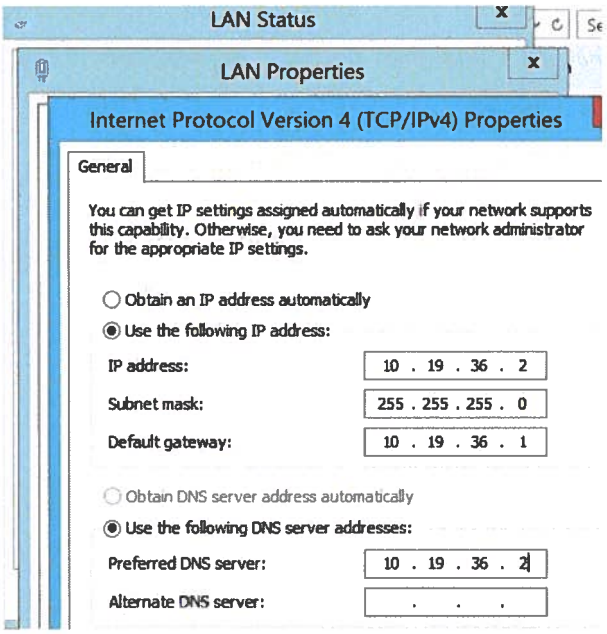

Figura 18 – Configuração do endereço IP.

Neste momento o servidor encontra-se com dois adaptadores de rede um para acesso LAN (Local) e outro WAN[7] (Internet).

Torna-se necessário efetuar configurações que obriguem a que as aplicações comuniquem por defeito pela rede de tipo "*host-only*" e não pela NAT. Para isso tem de se definir a ordem de prioridade do adaptador de rede a ser utilizado pelas aplicações.

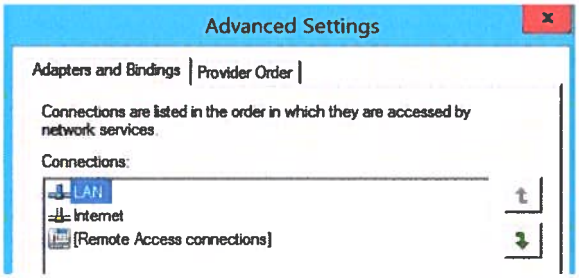

Figura 19 – Definição da prioridade do adaptador de rede.

Para evitar que as aplicações utilizem o adaptador de rede NAT tem de se impedir que este se registe no serviço DNS. Torna-se, assim, necessário remover a opção "*Register this connection's addresses in DNS*" no adaptador de rede NAT que foi designado pelo nome "*Internet*".

[7] *Wide Area Network* (Rede de Área Alargada)

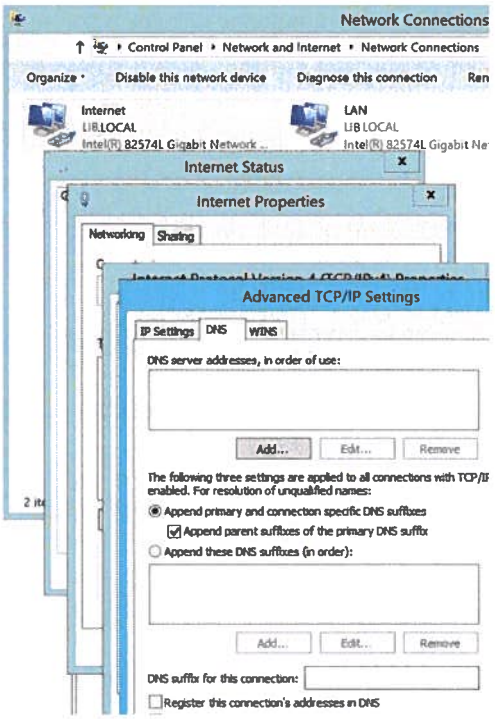

Figura 20 – Como impedir que o adaptador de rede se registe no DNS.

A segunda etapa é proceder à atualização do sistema operativo dos servidores. Para isso tem de se validar que o adaptador de rede NAT está ativado nas definições das máquinas virtuais.

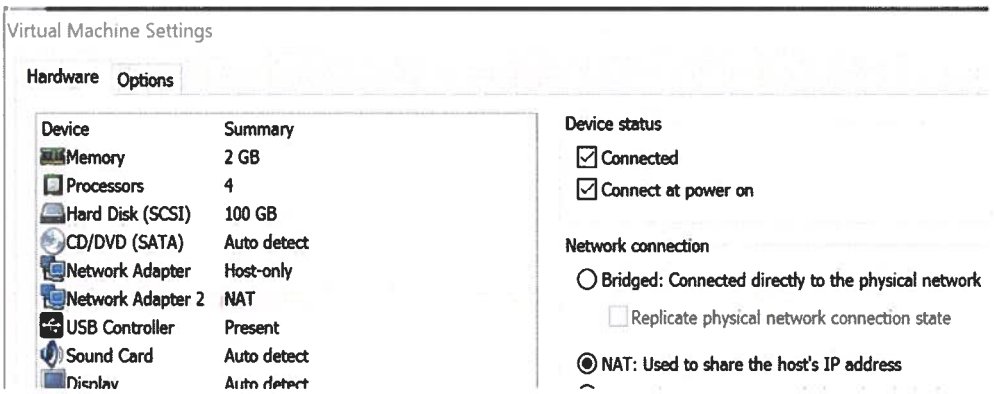

Figura 21 – Ativação do adaptador de rede NAT.

O adaptador de rede NAT das máquinas virtuais é automaticamente configurado com um endereço IP dinâmico que será utilizado para o acesso à Internet.

A atualização do sistema operativo faz-se através do "*windows updates*" que se encontra no painel de controlo. Uma vez atualizado, será necessário reiniciar o servidor e validar se continuam a haver novas atualizações por instalar.

6.5. Promoção do servidor DC1 a controlador de domínio

O servidor DC1 será usado como controlador de domínio, pelo que terá de se proceder à instalação da *role* "*Active Directory Domain Services*" e, de seguida, promover o servidor a controlador de domínio. Neste caso será necessário efetuar os seguintes comandos PowerShell:

- **Add-WindowsFeature AD-Domain-Services –IncludeManagementTools**
- **Install-ADDSForest –DomainName "LIB.LOCAL"**

Podemos ver o resultado destes comandos na

Figura 22.

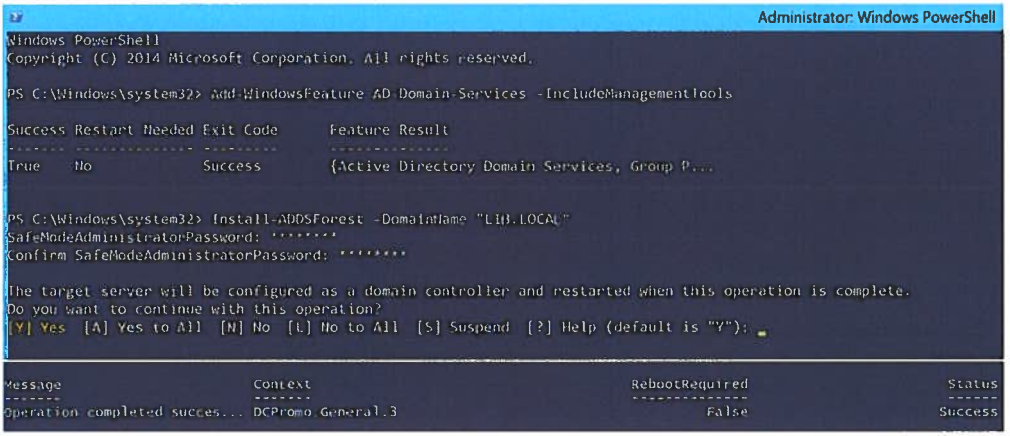

Figura 22 – Instalação da role ADDS e promoção a DC.

Como se pode verificar na

Figura 22 apenas foi preciso especificar o nome do domínio e a *password* do administrador; as restantes configurações do controlador de domínio serão assumidas por defeito.

É de notar que o serviço DNS é instalado como pré-requisito para dar suporte ao domínio "LIB.LOCAL".

6.6. Instalação e configuração da role DHCP[8] Server no DC1

Este serviço é importante para redes que contenham vários computadores.

Como o trabalho se prende a uma rede que tem um número muito limitado de computadores, não seria necessário utilizar-se o serviço DHCP, no entanto, pretende-se demonstrar a sua instalação que é efetuada executando o seguinte comando:

- Add-WindowsFeature -IncludeManagementTools dhcp

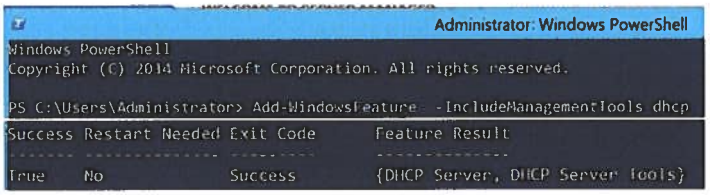

Figura 23 – Instalação da role DHCP.

De seguida é criado um scope (Figura 24) ao qual se especifica um nome e adiciona um range de IPs, assim como outras configurações de rede opcionais, que serão atribuídos dinamicamente aos computadores que estejam ligados na rede.

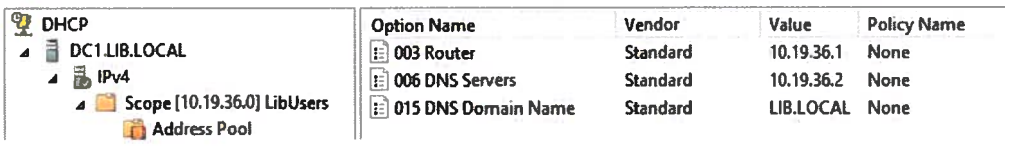

Figura 24 – Configurações do serviço DHCP.

[8] Dynamic Host Configuration Protocol

6.7. Servidores adicionados ao domínio "LIB.LOCAL"

Após a instalação, atualização e configuração de todos os servidores, é altura de adicionar o servidor HV1 e SQL ao domínio "LIB.LOCAL" (

Figura 25).

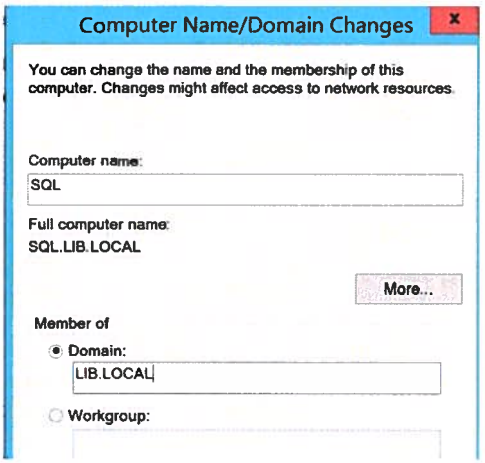

Figura 25 – Adicionar HV1 e SQL ao domínio LIB.LOCAL.

Ambos os servidores serão reiniciados.

6.8. Criação de *users* e grupos no Active directory

A autenticação nos servidores deve ser efetuada com uma conta de administração pessoal atribuída a cada administrador e não com uma conta genérica.

Os acessos de utilizadores privilegiados, a cada um dos servidores, devem ser atribuídos a um grupo de domínio e não diretamente a grupos locais de cada servidor.

Assim decidiu-se criar as seguintes contas e grupos com os seguintes privilégios:

Tabela 4 – Contas de utilizadores privilegiados

Display Name	*Username*	Membro de grupo	Privilégios
Rui Araujo ADM	LIB\raraujoadm	LIB\Domain Admins	Administrador de domínio
Pedro Silva ADM	LIB\psilvaadm	LIB\ALL_ServersAdmins	Administrador local de servidores
André Antunes ADM	LIB\aantunesadm	LIB\SQL Server Admins	Administrador de SQL

Vitor Pinto ADM	LIB\vpintoadm	LIB\SCVMMAdmins	Administrador do SCVMM
Pedro Pires	LIB\ppires	LIB\administradores Biblioteca	Administradores da Biblioteca

Por questões de simplificação na apresentação deste projeto irá utilizar-se uma conta genérica que terá todos os privilégios.

Tabela 5 – Conta genérica com todos os acessos privilegiados

Display Name	*Username*	Membro de grupo	Privilégios
SCVMM Admin	LIB\scvmm_admin	LIB\Domain Admins LIB\ALL_ServersAdmins LIB\SQL Server Admins LIB\SCVMMAdmins	Administrador de domínio Administrador de servidores Administrador de SQL Administrador do SCVMM

Foi também criada a conta de utilizador LIB\pfonseca (Pedro Fonseca) que será utilizada para reproduzir a experiência de utilizador na interação com a aplicação.

Este utilizador foi adicionado ao grupo LIB\utilizadores Biblioteca de modo a ter permissão para aceder à base de dados.

Tabela 6 – Especificações da conta de utilizador de Pedro Fonseca

Display Name	*Username*	Membro de grupo	Privilégios
Pedro Fonseca	LIB\pfonseca	LIB\Utilizadores Biblioteca	Utilizador da aplicação biblioteca virtual

Mais tarde irão ser criadas as contas de serviços para serem utilizadas nas configurações do SQL e do SCVMM. Assim que a base de dados SQL estiver construída, pode atribuir-se as permissões de acesso à base de dados aos utilizadores e administradores da aplicação.

7. Instalação e configuração do *System Center Virtual Machine Manager*

A Biblioteca irá disponibilizar as máquinas virtuais através da plataforma de virtualização Hyper-V. Esta plataforma de virtualização, permite que um administrador crie uma máquina virtual com o sistema operativo Windows 8.1 e com um elevado nível de customização em poucos minutos. Assim, quando um utilizador da biblioteca solicitar uma máquina virtual, este poderá imediatamente utilizá-la nas suas pesquisas.

O procedimento de instalação e configuração desta plataforma de virtualização irá ser descrito em seguida.

7.1. Instalação do SQL server no servidor com nome "SQL"

Para efetuar a instalação do SCVMM, existem alguns pré-requisitos de *software* que se deverão ter em conta e o SQL server é um deles.

Na instalação do SQL server há a necessidade de utilizar a conta de serviço Lib\scv_sql com a qual os serviços serão iniciados.

O SQL é instalado com as *features* "Database Engine Services" e "Management Tools – Complete" (Figura 26).

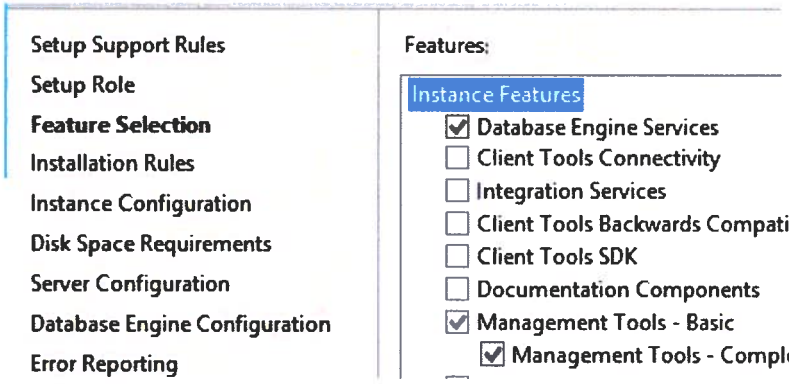

Figura 26 – SQL Instance Features.

A instância default MSSQLSERVER é mantida (

Figura 27).

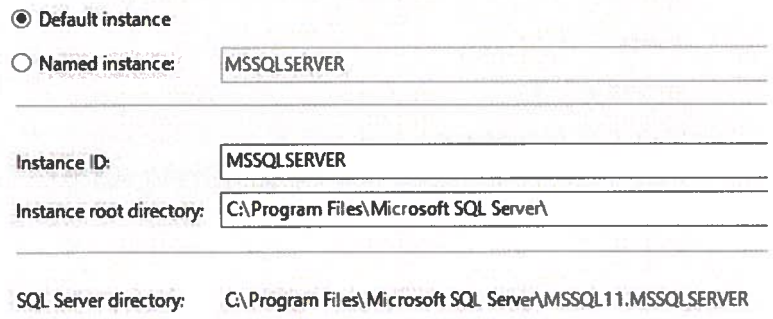

Figura 27 - Instância Default do SQL.

A conta de serviço scv_sql que será utilizada para iniciar os serviços SQL (Figura 28) é configurada.

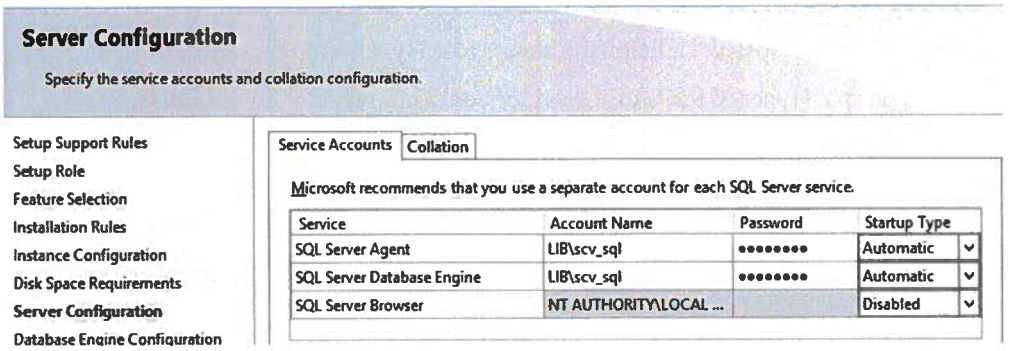

Figura 28 – Configuração da conta de serviço SQL.

De seguida é apresentada uma janela onde se define as contas e grupos que terão privilégios de administração sobre a instância de SQL. Neste ponto foi decidido adicionar apenas o grupo de domínio LIB\SQL Server Admins que havia sido criado anteriormente.

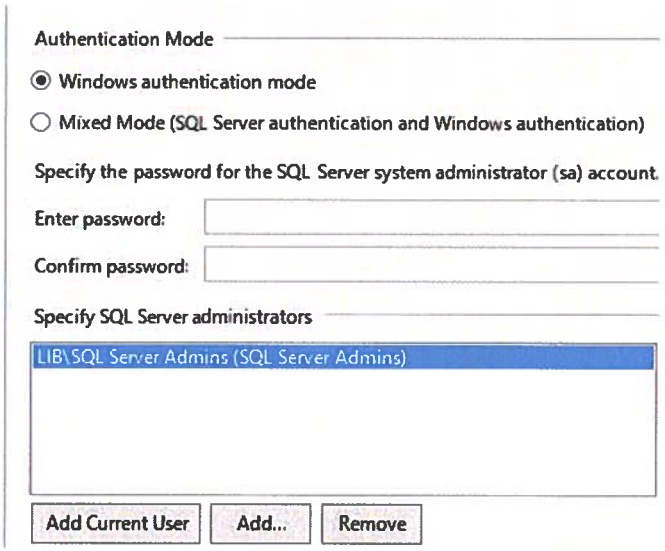

Figura 29 – Modo de autenticação Windows.

De seguida o processo de instalação termina e o SQL fica instalado.

7.2. Role Hyper-V e instalação do SCVMM no servidor HV1

Nesta altura já é possível adicionar a server role Hyper-V no servidor HV1. A instalação da role Hyper-V torna este servidor como um servidor de virtualização do tipo 1 (Figura 30).

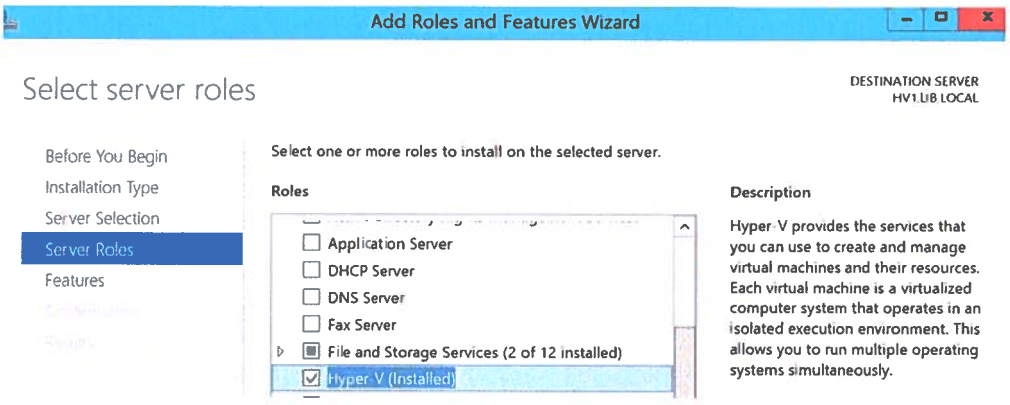

Figura 30 – Adicionar role Hyper-V.

Como pré-requisito efetuou-se a instalação do ADK (*Windows Assessment and Deployment Kit* para o Windows 8.1), selecionando apenas as opções "Deployment tools" e "Windows PE" (Figura 31).

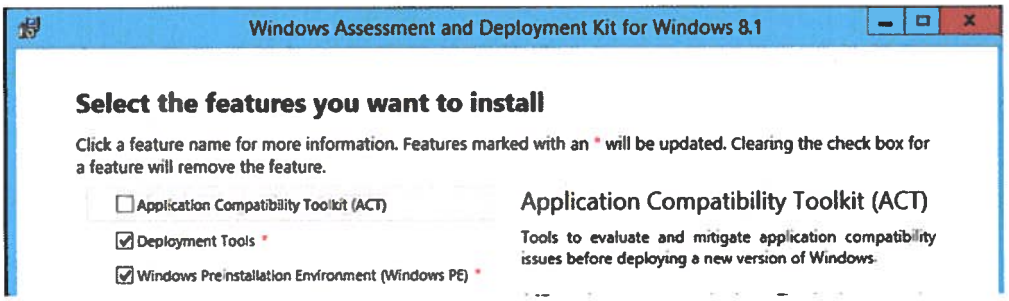

Figura 31 – Instalação do ADK 8.1.

Para que na instalação do SCVMM seja possível haver comunicação com a base de dados SQL abriram-se as portas TCP e UDP 1433 na Firewall do servidor SQL (Figura 32).

Figura 32 – Regra de Firewall.

A instalação do SCVMM foi iniciada a partir do CD de instalação e definiram-se as funcionalidades que se pretendiam atribuir na instalação (Figura 33).

Figura 33 – Funcionalidades do SCVMM.

Após, especificaram-se as configurações necessárias para a criação da base de dados.

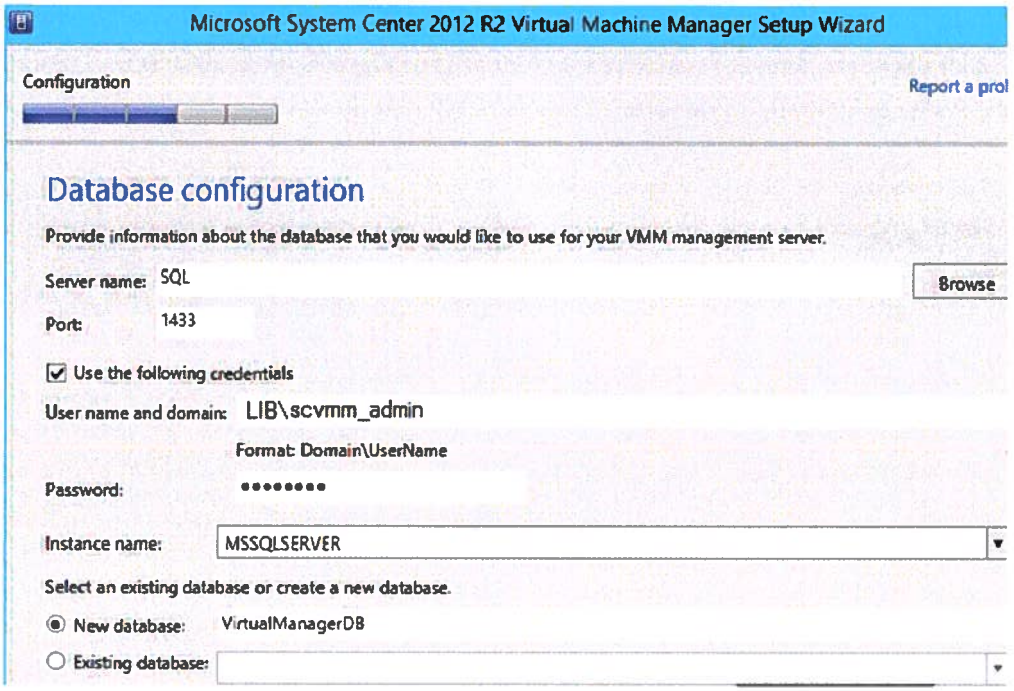

Figura 34 – Configuração da base de dados.

Aqui foi configurada a conta de serviço utilizada pelo VMM e a gestão da distribuição de chaves (Figura 35).

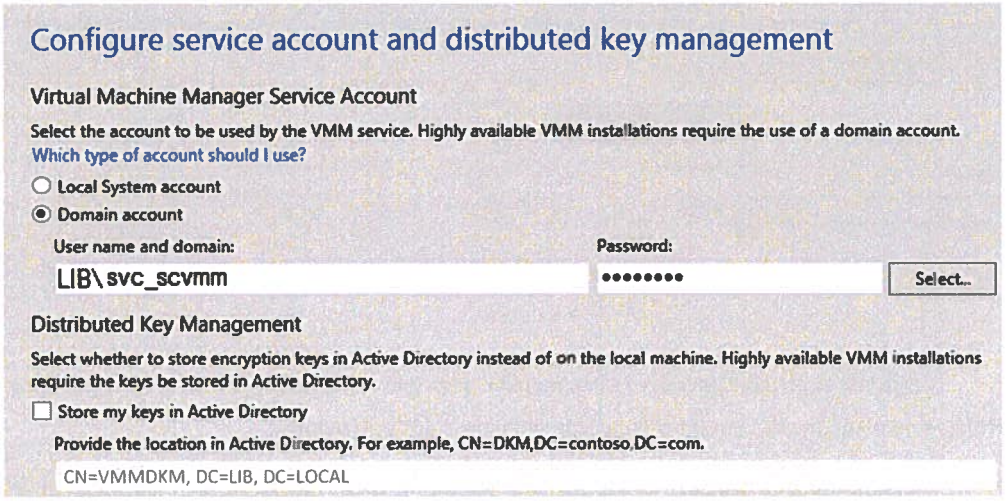

Figura 35 – Configuração da conta de serviços.

De seguida o processo de instalação termina e o SCVMM fica instalado.

7.3. Configuração do System Center VMM

Utilizando a consola de administração do SCVMM (Figura 36), efetua-se a configuração da rede de produção que será utilizada pelas máquinas virtuais.

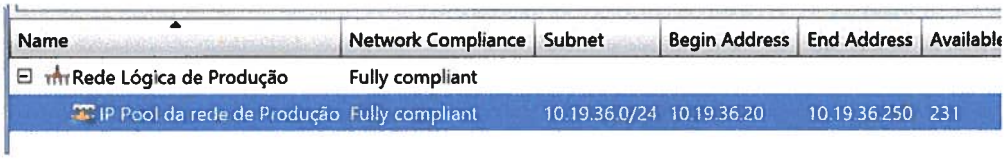

Figura 36 – IP Pool da rede de produção.

Os recursos de *hardware* a serem atribuídos às máquinas virtuais dos utilizadores da biblioteca serão sempre os mesmos, pelo que faz todo o sentido criar um perfil de *hardware*. O administrador da biblioteca terá apenas que escolher este perfil já com as definições pré-definidas do *hardware* a atribuir à máquina virtual.

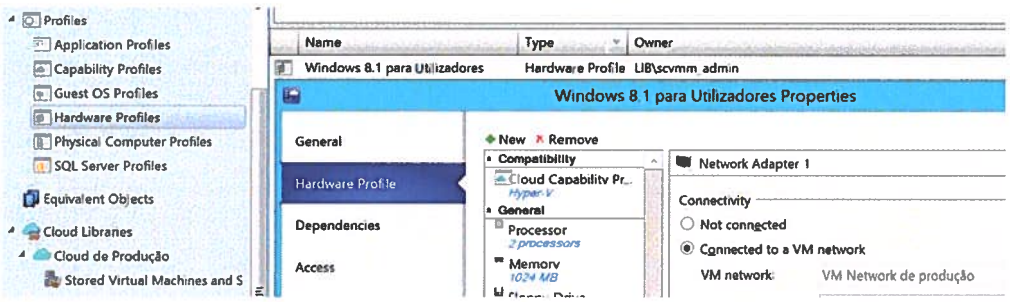

Figura 37 – Perfil de hardware.

Os perfis de sistema operativo são essenciais para que a configuração das novas máquinas virtuais sejam atualizadas em função das características dos utilizadores.

Estes perfis apresentados na Figura 37 e Figura 38 serão utilizados pelo administrador sempre que se pretenda criar uma nova máquina virtual para os utilizadores da biblioteca.

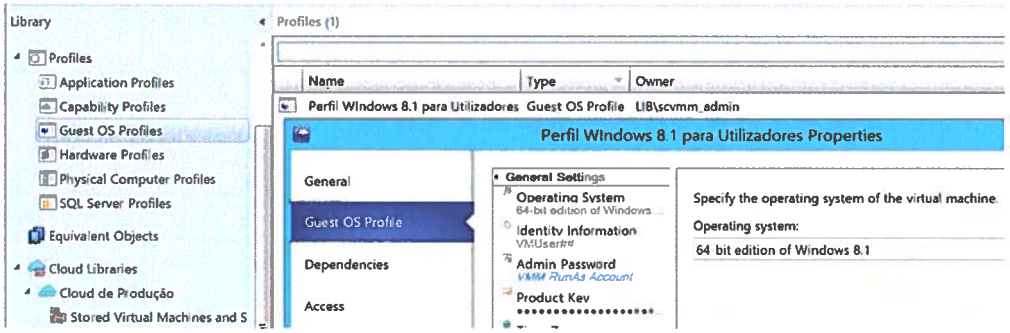

Figura 38 – Perfil do sistema operativo.

Desta forma, foi criada uma máquina virtual com a utilização dos perfis que foram anteriormente criados nos quais ficou pré-definido o sistema operativo Windows 8.1 assim como outras características. Mais tarde esta máquina será convertida em VM template.

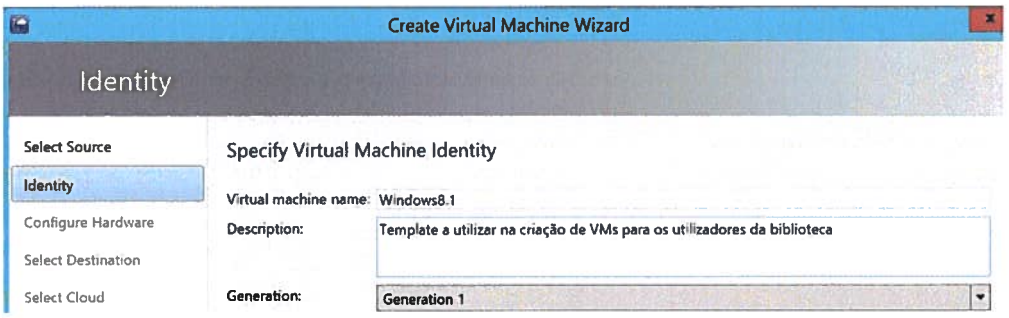

Figura 39 – Criação da primeira máquina virtual.

Todas as novas máquinas virtuais irão passar a ser criadas a partir de um VM *template*.

Para isso tem de converter em *template* a máquina virtual criada anteriormente.

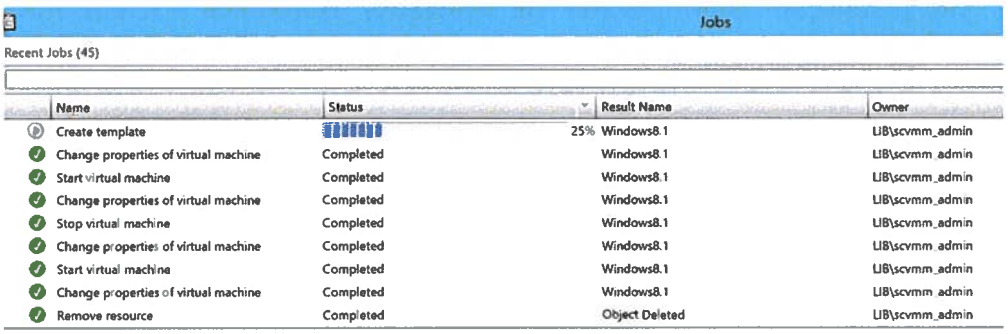

Figura 40 – Criação da VM Template.

Nesta fase todo o sistema está apto para disponibilizar máquinas virtuais aos utilizadores, no entanto estas não terão utilidade prática na biblioteca enquanto a aplicação não estiver instalada.

Para a criação das máquinas virtuais a partir da *template*, o administrador apenas terá de selecionar "*VM templates*" e "*create virtual machine*".

Nos próximos passos será apenas necessário escolher os perfis de *hardware* e sistema operativo que já incluem as configurações predefinidas.

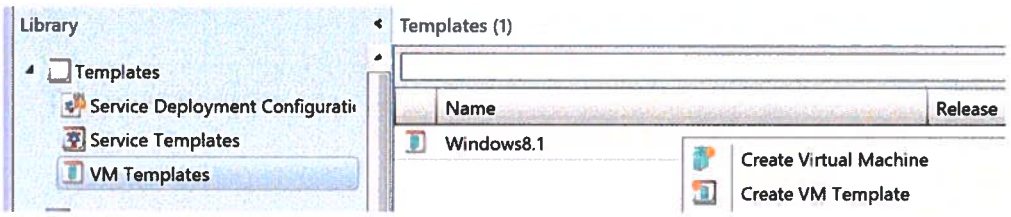

Figura 41 – Criação da máquina virtual a partir da template.

Esta máquina virtual criada para um utilizador terá agora de ser configurada com as aplicações necessárias para que o utilizador tenha acesso aos livros e outros documentos disponíveis na base de dados da biblioteca virtual.

Também o administrador terá acesso à aplicação para que possa atualizar a base de dados com novos conteúdos, como mencionado anteriormente.

7.4. Instalação automatizada da aplicação Biblioteca Virtual nos postos de trabalho

Os utilizadores irão ter acesso a uma aplicação Windows que lhes será disponibilizada na máquina virtual. A aplicação será acedida por um atalho que se encontra no ambiente de trabalho.

A aplicação Windows é distribuída pelas novas máquinas virtuais com base na *Group policy* aplicada à OU onde se encontra a conta que foi criada para o utilizador (Figura 42).

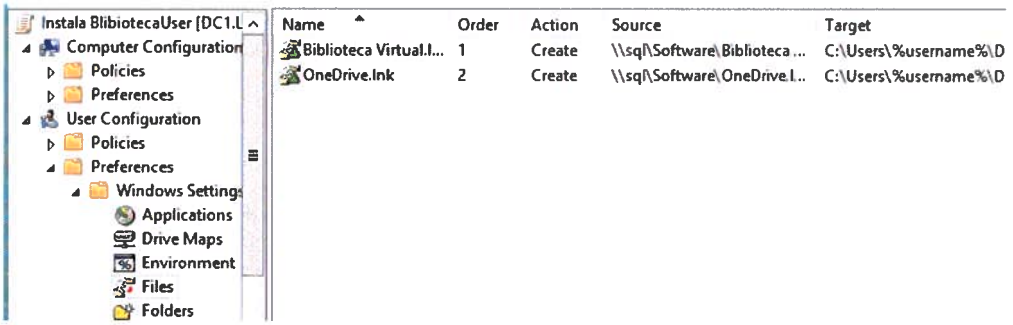

Figura 42 – Group policies para distribuição da aplicação.

Existe uma GPO, associada à OU de administradores da biblioteca e uma outra GPO associada à OU de utilizadores da biblioteca (Figura 43).

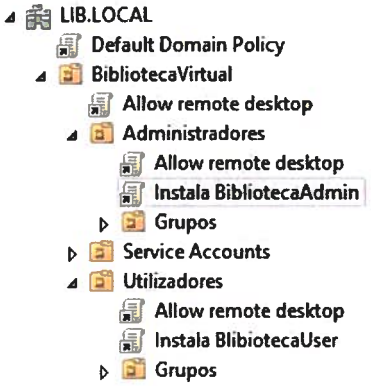

Figura 43 – Group policies para instalação da aplicação.

Quando o administrador ou utilizador da biblioteca efetuam *login* no posto de trabalho virtual, a GPO é aplicada fazendo com que a aplicação Windows (Biblioteca virtual) de administração da base de dados, ou de utilização, seja disponibilizada no posto de trabalho, por intermédio de um atalho.

8. Interface da aplicação Biblioteca Virtual

8.1. Interface para o utilizador

A interface da aplicação para o utilizador da biblioteca é muito intuitiva e a sua utilização é muito simples. O utilizador poderá pesquisar e consultar publicações, assim como transferir essas publicações para o seu posto de trabalho ou para a sua conta na *Cloud* da Microsoft.

De seguida demostra-se como o utilizador poderá pesquisar e consultar as publicações.

Ao executar a aplicação é apresentada ao utilizador uma janela onde poderá efetuar a pesquisa (Figura 44).

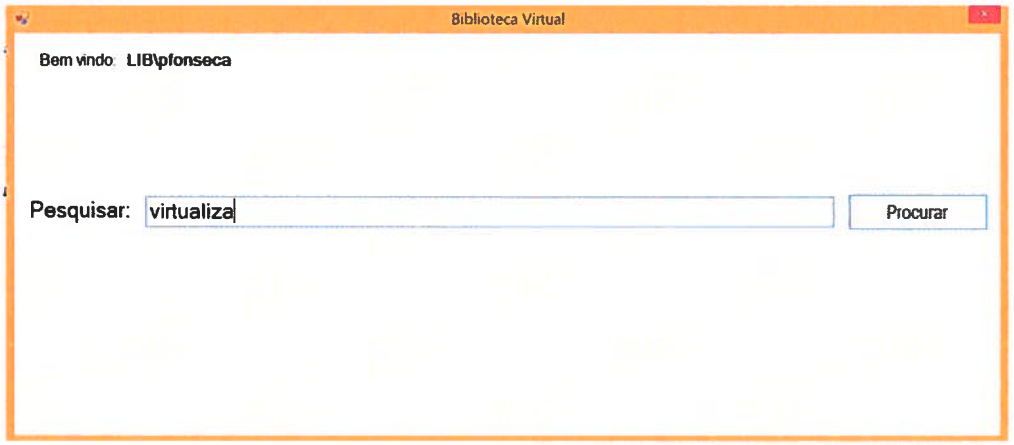

Figura 44 – Janela de Pesquisa da Biblioteca Virtual.

O resultado da pesquisa é apresentado numa janela (Figura 45) onde são disponibilizadas opções avançadas de pesquisa. O utilizador poderá assim utilizar critérios de pesquisa ainda mais específicos.

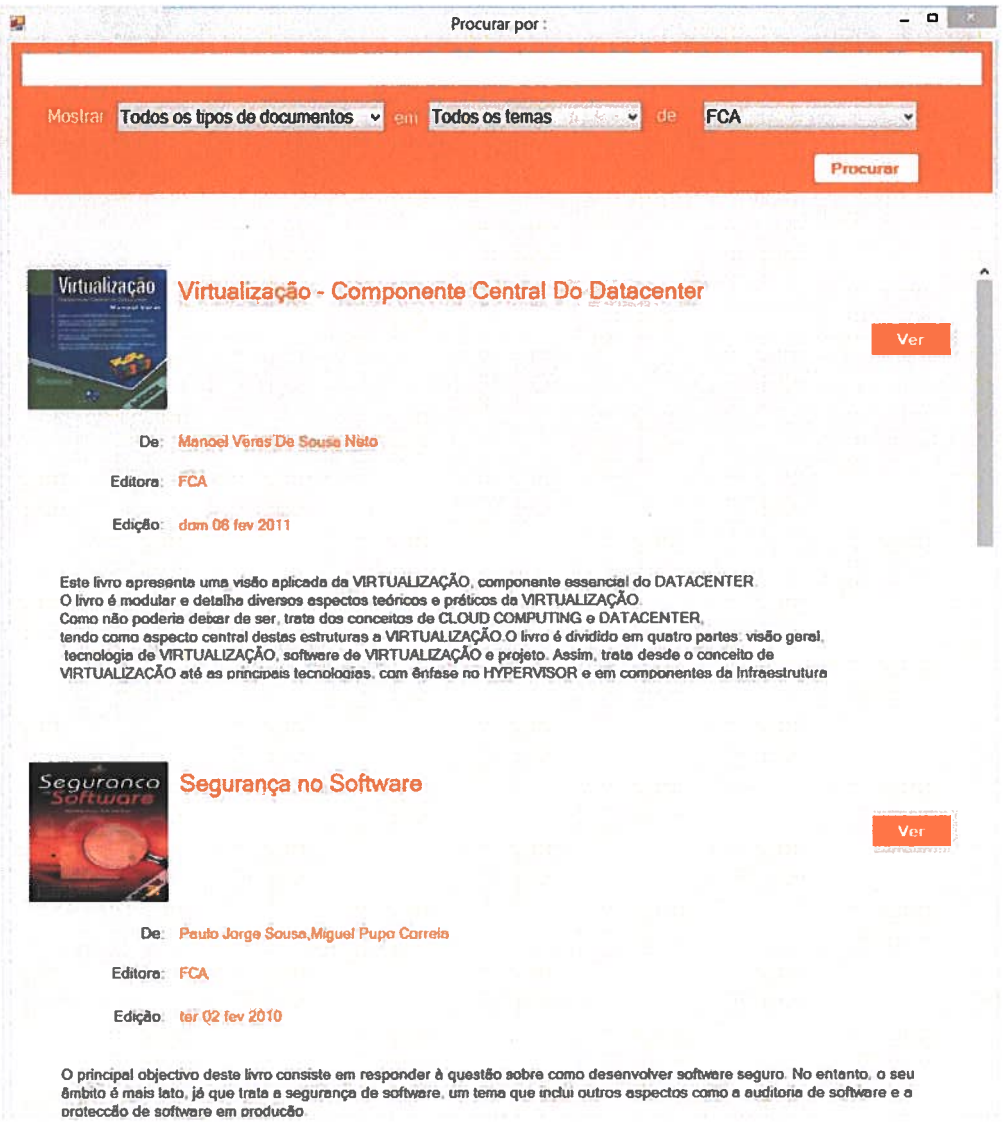

Figura 45 – Resultado de Pesquisa na Biblioteca Virtual.

Ao selecionar o botão Ver o utilizador tem a possibilidade de ler a sinopse e a opção para transferir o documento em PDF para o seu posto de trabalho (Figuras 45 e 46).

Figura 46 – Detalhes de Pesquisa.

O utilizador poderá guardar todos os documentos consultados na *Cloud* Microsoft.

Para que o utilizador transfira a publicação para a sua conta, terá de executar a aplicação "Microsoft Cloud" que se encontra como atalho no seu posto de trabalho. Nesse instante será solicitado que introduza as credenciais de autenticação. Após a autenticação com sucesso o utilizador terá acesso à pasta *OneDrive*, onde poderá colocar os documentos que queira rever mais tarde.

8.2. Interface para o administrador

A interface da aplicação para o administrador é ligeiramente mais complexa que a do utilizador, mas foi desenvolvida para ser de fácil utilização. Foi desenhada para a gestão de livros na base de dados permitindo adicionar novos livros, alterar ou apagar.

A atualização da informação sobre livros e autores pode ser feita muito rapidamente, pois toda a informação sobre o livro a criar ou atualizar encontra-se acessível na mesma janela da aplicação.

De seguida irá demonstrar-se como o administrador poderá adicionar, atualizar ou eliminar um livro na base de dados.

Para ter acesso à aplicação, o administrador acede remotamente a uma máquina virtual que foi disponibilizada pelo administrador do SCVMM. Efetua a autenticação com as credenciais que lhe foram fornecidas (Figura 47). O acesso à aplicação será disponível por um atalho criado no ambiente de trabalho. Ao executar esta aplicação é-lhe apresentada a janela onde terá de introduzir as suas credenciais.

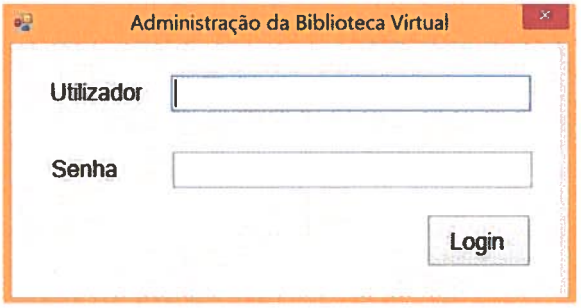

Figura 47 – Janela de Credenciais.

Na página inicial estão listados os livros existentes e do lado direito são apresentados os acessos dos usuários, com data e hora, conforme se vê na Figura 48.

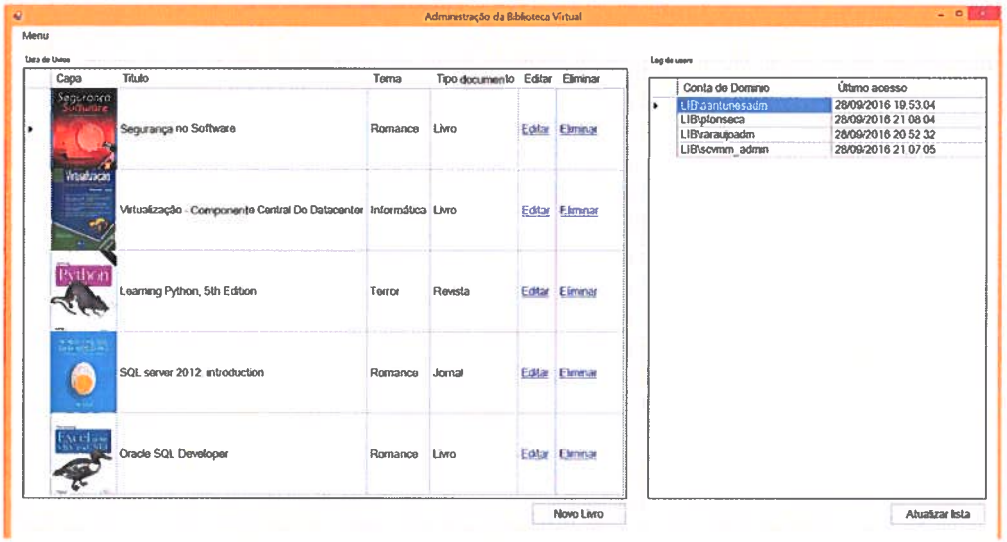

Figura 48 – Página Inicial da Biblioteca Virtual.

As funções dos botões existentes nesta janela são as seguintes:

"**Editar**" – Altera os conteúdos dos livros

"**Eliminar**" – Apaga o livro

"**Novo livro**" – Adiciona um novo livro

"**Atualizar lista**" – Atualiza os acessos dos utilizadores

"**Menu**" – Adiciona as áreas temáticas dos livros, altera a senha e sai da aplicação.

Quando o administrador pretende adicionar livros na base de dados é-lhe apresentada a janela que se pode ver na Figura 49.

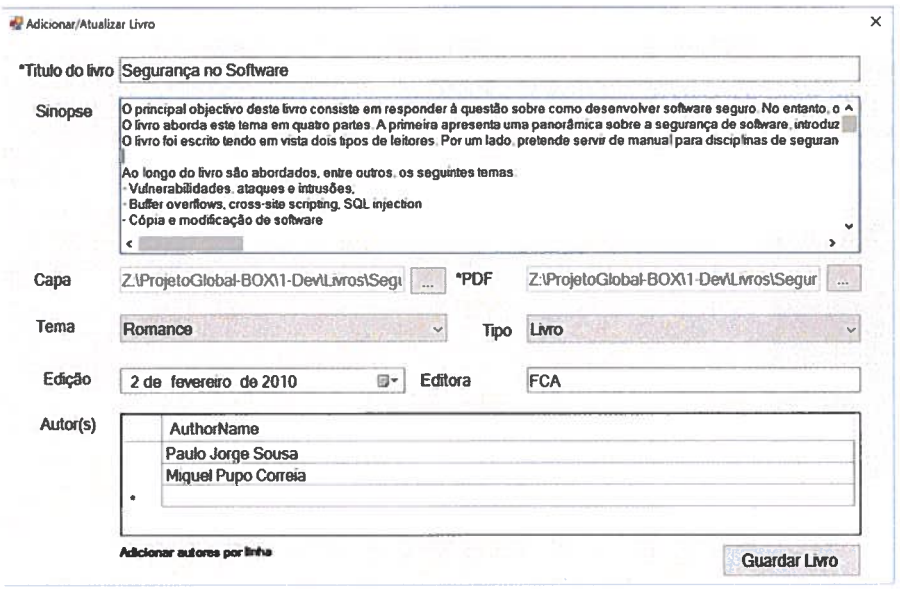

Figura 49 - Janela de Atualização/Adição de Livro.

Adicionar ou atualizar um livro na base de dados passa por preencher ou alterar os campos existentes com os dados referentes a esse livro. Estes dados são os seguintes:

Título do livro

Sinopse – Breve descrição sobre o conteúdo do livro

Capa – Imagem da capa

PDF File – Ficheiro digitalizado.

Tema – Tema com que o conteúdo deste livro se relaciona

Tipo – Tipo de publicação.

Edição – Data em que o livro foi publicado

Editora – Entidade que publicou o livro

Author(s) – Pessoa ou pessoas que escreveram o livro.

Os livros podem ser também eliminados.

Na área de administração o utilizador tem ainda um menu onde pode adicionar as áreas temáticas dos livros, alterar a senha e sair, da aplicação (Figura 50).

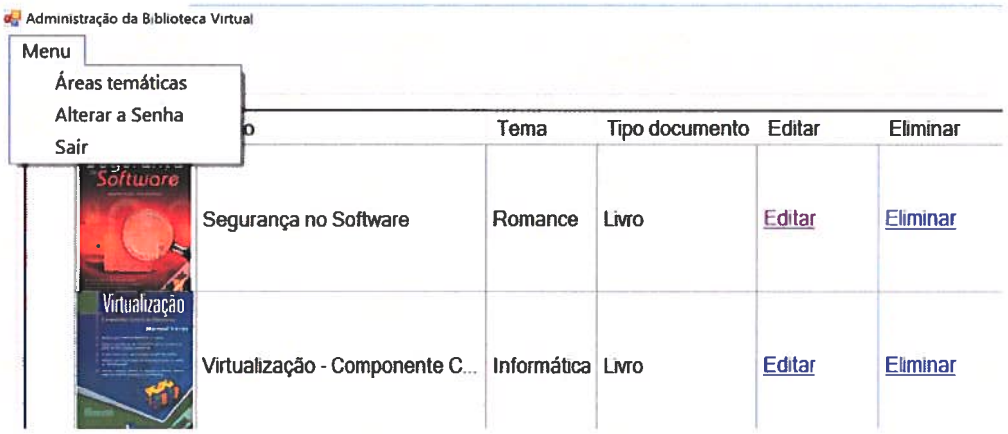

Figura 50 – Inserir Áreas Temáticas.

9. Desenho e implementação da base de dados

A aplicação Biblioteca Virtual necessita de uma base de dados onde serão colocados todos os conteúdos relacionados com publicações, autores, utilizadores e outras informações adicionais, tais como data de publicação, nome do livro, hora de login, etc.

9.1. Diagrama da base de dados

A base de dados Biblioteca virtual foi criada com 6 tabelas. Na tabela *VMUsers* tem guardada informação sobre o utilizador e a data de login (Figura 51). Na tabela *SystemAdmins* estão guardadas as credenciais dos administradores da Biblioteca. A tabela Books está relacionada com as tabelas *BookCategories*, *BookAuthors* e *DocumentTypes* e é onde se encontra toda a informação sobre os livros.

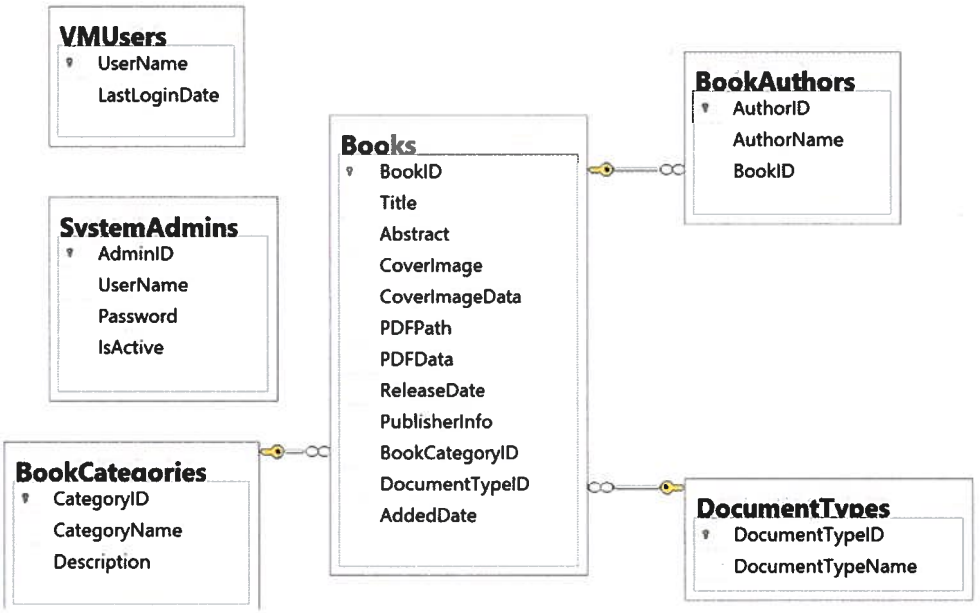

Figura 51 – Tabela Relacional SQL.

9.2. Componentes da BD (Views, storeprocedures, Tabelas, Foreign keys)

Adicionalmente às tabelas foi construída uma *View* (Figura 52) para as tabelas relacionadas com livros.

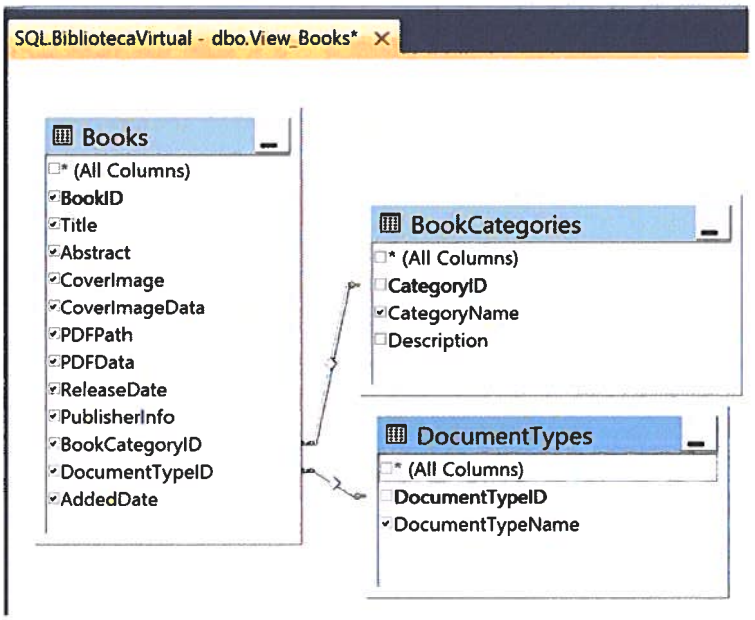

Figura 52 – View_Books.

Código da View view_books

```
SELECT dbo.Books.BookID, dbo.Books.Title, dbo.Books.Abstract, dbo.Books.CoverImage, dbo.Books.CoverImageData,
dbo.Books.PDFPath, dbo.Books.PDFData, dbo.Books.ReleaseDate, dbo.Books.PublisherInfo, dbo.Books.BookCategoryID,
dbo.Books.DocumentTypeID, dbo.Books.AddedDate, dbo.BookCategories.CategoryName,

    dbo.DocumentTypes.DocumentTypeName

FROM  dbo.Books INNER JOIN

    dbo.BookCategories ON dbo.Books.BookCategoryID = dbo.BookCategories.CategoryID INNER JOIN

    dbo.DocumentTypes ON dbo.Books.DocumentTypeID = dbo.DocumentTypes.DocumentTypeID
```

Nesta tabela virtual (*view*) são listados os livros tendo em conta os vários campos relacionados nas várias tabelas (*Books, DocumentTypes* e *BookCategories*) presentes na Base de Dados.

Foram criadas as seguintes *stored procedures* (Figura 53), de forma a poder facilitar o acesso aos conteúdos da base de dados no desenvolvimento da aplicação.

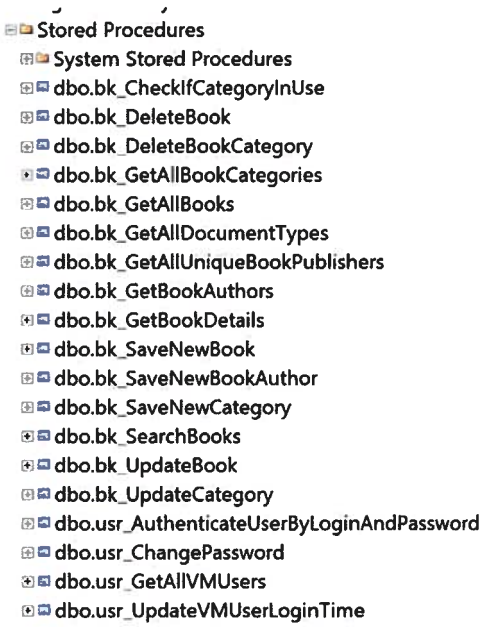

Figura 53 – Stored Procedures.

Em seguida irá ser descrito o objetivo de cada stored procedure. O código da *Stored Procedure* na Figura 54 permite-nos validar quantas vezes uma determinada categoria é utilizada na tabela books.

```
ALTER Procedure [dbo].[bk_CheckIfCategoryInUse]
@CategoryID int

AS

 Select count(BookID) as BooksCount from Books where BookCategoryID=@CategoryID
```

Figura 54 – bk_CheckIfCategoryInUse.

A stored procedure bk_DeleteBook na Figura 55 efetua a eliminação do livro e do seu autor das tabelas livros e autores.

```
ALTER Procedure [dbo].[bk_DeleteBook]
@BookID bigint

AS

Delete from BookAuthors  where BookID=@BookID

 Delete from Books

where
    BookID=@BookID
```

Figura 55 – bk_DeleteBook.

A *Stored Procedure* bk_DeleteBookCategory (Figura 56) elimina a categoria que, por exemplo, possa ter sido inserida por erro ou que já não seja utilizada.

```
ALTER Procedure [dbo].[bk_DeleteBookCategory]
@CategoryID int

AS

Delete from BookCategories
where
    CategoryID=@CategoryID
```

Figura 56 – bl_DeleteBookCategory.

A *Stored Procedure* bk_GetAllBookCategories (Figura 57), obtém todas as categorias de livros (livro de romance, terror,...).

```
ALTER Procedure [dbo].[bk_GetAllBookCategories]

AS

sELECT * FROM BookCategories
```

Figura 57 – bk_GetAllBookCategories.

A *stored procedure* bk_GetAllBooks (Figura 58) permite obter todas as características que estão presentes na View, View_Books e ordená-las por ordem decrescente pela data em que os livros foram inseridos.

```
ALTER Procedure [dbo].[bk_GetAllBooks]

AS

sELECT * FROM View_Books order by AddedDate desc
```

Figura 58 – bk_GetAllBooks.

Algumas das colunas com a informação obtida por essa *procedure* é apresentada na Figura 59.

	BookID	Title	Abstract	PublisherInfo	ReleaseDate	CoverImage
1	8	Learning Python, 5th Edition	Book Description Get a comprehensive, in-depth in...	O'Reilly Media, Inc	2016-06-10	C:\Users\ibadmin\Desktop\transferirpython.jpg
2	7	SQL server 2012 introduction	How do we design for data when traditional design t...	Porto editora	2016-06-09	C:\Users\ibadmin\Desktop\transferirsql.jpg
3	6	NoSQL and SQL Data Modeli...	How do we design for data when tr aditional design...	Technics Publications	2016-04-05	C:\Users\ibadmin\Desktop\transferirsql.jpg
4	5	Oracle SQL Developer	You are expected to have programming knowledge ...	Packt Publishing	2016-01-05	C:\Users\ibadmin\Desktop\New folder\transferir
5	4	book cover test	book cover test	Book test	2016-06-08	C:\Users\ibadmin\Desktop\transferir.jpg

Figura 59 – Resultado da Procedure bk_GetAllBooks.

A *stored procedure* bk_GetAllDocumentTypes (Figura 60) obtém todos os tipos de documentos existentes consoante o seu género (livro, jornal, imagem, manuscrito).

```
ALTER Procedure [dbo].[bk_GetAllDocumentTypes]

AS

sELECT * FROM DocumentTypes
```

Figura 60 – bk_GetAllDocumentTypes.

A *Stored procedure* bk_GetAllUniqueBookPublishers (Figura 61) obtém a lista de todas as editoras existentes na tabela *books*, onde necessariamente o campo *PublisherInfo* é maior que 0.

```
ALTER Procedure [dbo].[bk_GetAllUniqueBookPublishers]

AS
  Select Distinct PublisherInfo from Books
  where Len(PublisherInfo) > 0
```

Figura 61 – bk_GetAllUniqueBookPublishers.

A *stored procedure* bk_Get_BookAuthors (Figura 62) obtém a lista de todos os autores (por nome e ID) existentes na tabela BookAuthors.

```
ALTER Procedure [dbo].[bk_GetBookAuthors]
@BookID bigint

AS

sELECT AuthorID,AuthorName FROM BookAuthors where BookID=@BookID
```

Figura 62 – bk_GetBookAuthors.

A *stored procedure* bk_GetBookDetails (Figura 63) obtém a lista dos autores (pode ser mais que um) de um determinado livro que se encontra inserido na Base de Dados, utilizando a função fn_GetBookAuthorsCSV.

```
ALTER Procedure [dbo].[bk_GetBookDetails]
 @BookID bigint
AS

sELECT *,dbo.fn_GetBookAuthorsCSV(@BookID) as Authors FROM View_Books where BookID=@BookID
```

Figura 63 – bk_GetBookDetails.

A *stored procedure* bk_SaveNewBook (Figura 64) permite fazer a criação de um novo livro, onde são atualizados todos os campos visíveis, aquando a criação do mesmo, através da interface da aplicação com o perfil de administração.

```
ALTER Procedure [dbo].[bk_SaveNewBook]
@BookTitle varchar(250),
@Abstract varchar(5000),
@CoverImage varchar(500),
@CoverImageData varbinary(max),
@PDFPath varchar(500),
@PDFData varbinary(max),
@ReleaseDate date,
@PublisherInfo varchar(250),
@BookCategoryID bigint,
@DocumentTypeID int

AS

 INSERT INTO Books
 VALUES
        (@BookTitle,
        @Abstract,
        @CoverImage,
        @CoverImageData,
        @PDFPath,
        @PDFData,
        @ReleaseDate,
        @PublisherInfo,
        @BookCategoryID,
        @DocumentTypeID,
        GETDATE())

SELECT * from Books where BookID=@@IDENTITY
```

Figura 64 – bk_SaveNewBook.

Esta *stored procedure* (bk_SaveNewBookAuthor) permite associar um ou mais autor a um determinado livro, campos esses que poderão ser editados na interface gráfica da aplicação com o perfil de administração (Figura 65).

```
ALTER Procedure [dbo].[bk_SaveNewBookAuthor]
@AuthorName varchar(50),
@BookID bigint

AS

 INSERT INTO BookAuthors
 VALUES
        (@AuthorName,
        @BookID )
```

Figura 65 – bk_SaveNewBookAuthor.

Esta *procedure* bk_SaveNewCategory (Figura 66) permite fazer a criação das Categorias (Categorizar o livro consoante a sua temática) bem como a sua descrição na interface da aplicação em perfil de administração.

```
ALTER Procedure [dbo].[bk_SaveNewCategory]

@CategoryName varchar(50),
@Description varchar(500)

AS

INSERT INTO   BookCategories
 VALUES (@CategoryName, @Description)
```

Figura 66 – bk_SaveNewCategory.

Esta *stored procedure* bk_Searchbooks (Figura 67) tem como função devolver os dados com base nos termos de pesquisa, que poderá ser utilizando os campos disponíveis no módulo de pesquisa existente e acessível através do perfil de utilizador. Os campos de pesquisa que podem ser utilizados são:

- Categoria;
- Título;
- Autor;
- Tipo de Documento.

```
ALTER Procedure [dbo].[bk_SearchBooks]
@Title varchar(50),
@DocTypeID varchar(10),
@CategoryID varchar(10),
@PublisherInfo varchar(50)
AS
Declare @Select nvarchar(4000)
Declare @From nvarchar(100)
Declare @Where nvarchar(4000)
Declare @OrderBy nvarchar(150)
Declare @SQL nvarchar(4000)
select @Select = 'SELECT * ,dbo.fn_GetBookAuthorsCSV(BookID) as Authors'
select @From='FROM View_Books'
Set @Where=null
IF LEN(@Title) >0
BEGIN
Select @Where = 'Where Title like "%'+@Title+'%" or BookID in (Select BookID from BookAuthors where
AuthorName like "%'+@Title+'%")'
END
IF LEN(@DocTypeID) >0
BEGIN
Select @Where = Case When @Where is null then 'Where DocumentTypeID ='+@DocTypeID
Else @Where +'and DocumentTypeID ='+@DocTypeID
End
END
IF LEN(@CategoryID) >0
BEGIN
Select @Where = Case When @Where is null then 'Where BookCategoryID ='+@CategoryID
Else @Where +'and BookCategoryID ='+@CategoryID
End
END
IF LEN(@PublisherInfo) >0
BEGIN
Select @Where = Case When @Where is null then 'Where PublisherInfo ='"+@PublisherInfo+'"
Else @Where +'and PublisherInfo ='"+@PublisherInfo+'"
End
END
Set @SQL = @Select +' '+ @From +' '+ Coalesce(@Where,'') + ' '+ Coalesce(@OrderBy,'')
        exec (@SQL)
```

Figura 67 – Procedure bk_SearchBooks.

Esta *stored procedure* bk_Updatebook (Figura 68) permite fazer atualizações aos vários campos existentes num livro já inserido, sendo que estas apenas poderão ser realizadas através do perfil de administrador.

```
ALTER Procedure [dbo].[bk_UpdateBook]
@BookID bigint,
@BookTitle varchar(250),
@Abstract varchar(5000),
@CoverImage varchar(500),
@CoverImageData varbinary(max),
@PDFPath varchar(500),
@PDFData varbinary(max),
@ReleaseDate date,
@PublisherInfo varchar(250),
@BookCategoryID bigint,
@DocumentTypeID int

AS

 Update Books
 Set
         Title=@BookTitle,
        Abstract=@Abstract,
        CoverImage=@CoverImage,
        CoverImageData=@CoverImageData,
        PDFPath=@PDFPath,
        PDFData=@PDFData,
        ReleaseDate=@ReleaseDate,
        PublisherInfo=@PublisherInfo,
        BookCategoryID=@BookCategoryID,
        DocumentTypeID=@DocumentTypeID
where
    BookID=@BookID

Delete from BookAuthors  where BookID=@BookID
```

Figura 68 – bk_UpdateBook.

A *Stored Procedure* bk_UpdateCategory (Figura 69) permite fazer uma atualização às categorias já existentes; estas apenas poderão ser realizadas com permissão da administração.

```
ALTER Procedure [dbo].[bk_UpdateCategory]
@CategoryID int,
@CategoryName varchar(50),
@Description varchar(500)

AS

Update   BookCategories
set
    CategoryName=@CategoryName,
    Description=@Description
where
    CategoryID=@CategoryID
```

Figura 69 – bk_UpdateCategory.

Esta *stored Procedure* usr_AuthenticateUserByLoginAndPassword (Figura 70) permite fazer a autenticação à aplicação utilizando o nome de utilizador e *password.*

```
ALTER Procedure [dbo].[usr_AuthenticateUserByLoginAndPassword]
@UserName varchar(50),
@Password varchar(50)
AS

sELECT * FROM SystemAdmins where  UserName=@UserName  and  Password=@Password  and IsActive=1
```

Figura 70 – usr_AuthenticateUserByLoginAndPassword.

Esta *stored Procedure* usr_ChangePassword (Figura 71) permite fazer a alteração à palavra-chave do utilizador com privilégios de administração.

```
ALTER Procedure [dbo].[usr_ChangePassword]
@AdminID int,
@NewPassword varchar(50)
AS

Update SystemAdmins
set
    Password=@NewPassword
WHERE
    AdminID=@AdminID
```

Figura 71 – usr_ChangePassword.

Esta *stored Procedure* usr_GetAllVMUsers (Figura 72) permite fazer a listagem dos utilizadores, assim como a data e hora que entraram na aplicação, podendo esta informação ser visualizada na aplicação com o perfil de administração.

```
ALTER Procedure [dbo].[usr_GetAllVMUsers]

AS

sELECT * FROM VMUsers
```

Figura 72 – usr_GetAllVMUsers.

Esta *stored Procedure* usr_UpdateVMUserLoginTime (Figura 73) permite atualizar os dados relativamente aos registos de *login* de um utilizador.

```
ALTER Procedure [dbo].[usr_UpdateVMUserLoginTime]
@UserName varchar(50)
AS

if not exists(select UserName from VMUsers where UserName=@UserName)
BEGIN
    INSERT INTO VMUsers
    VALUES
        (@UserName,GETDATE())
END
ELSE
BEGIN
    UPDATE VMUsers
    SET
        LastLoginDate=GETDATE()
    WHERE
    UserName=@UserName
END
```

Figura 73 – usr_UpdateVMUserLoginTime.

Funções:

```
ALTER function [dbo].[fn_GetBookAuthorsCSV]( @BookID BIGINT)
   returns varchar(500)
as
BEGIN
      DECLARE @listStr VARCHAR(500)

    SELECT @listStr = COALESCE(@listStr+',' , '') + CAST([AuthorName] AS Varchar(50) )
    FROM BookAuthors where  BookID=@BookID
        RETURN @listStr
END
```

Figura 74 – fn_GetBookAuthorsCSV.

Em cima fica a construção da base de dados e a seguir tem-se as permissões a essa base de dados.

9.3. Permissões de acesso à base de dados

O grupo "LIB\Utilizadores Biblioteca" foi criado na AD e adicionado na base de dados SQL para controlar os acessos dos utilizadores que podem aceder à base de dados da biblioteca virtual, com permissões muito restritas. Os utilizadores membros deste grupo conseguirão autenticar-se com sucesso na aplicação e fazer pesquisas (Figura 75).

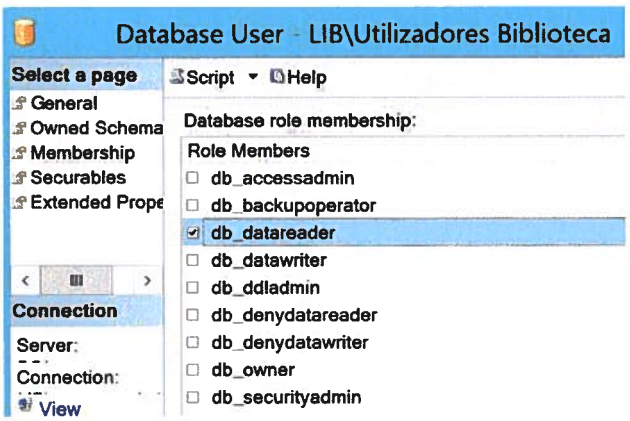

Figura 75 – LIB\Utilizadores Biblioteca.

Adicionalmente foi atribuída a permissão de execução, uma vez que o utilizador necessita de executar as *stored procedures* que foram criadas para auxiliar no desenvolvimento do codigol (Figura 76).

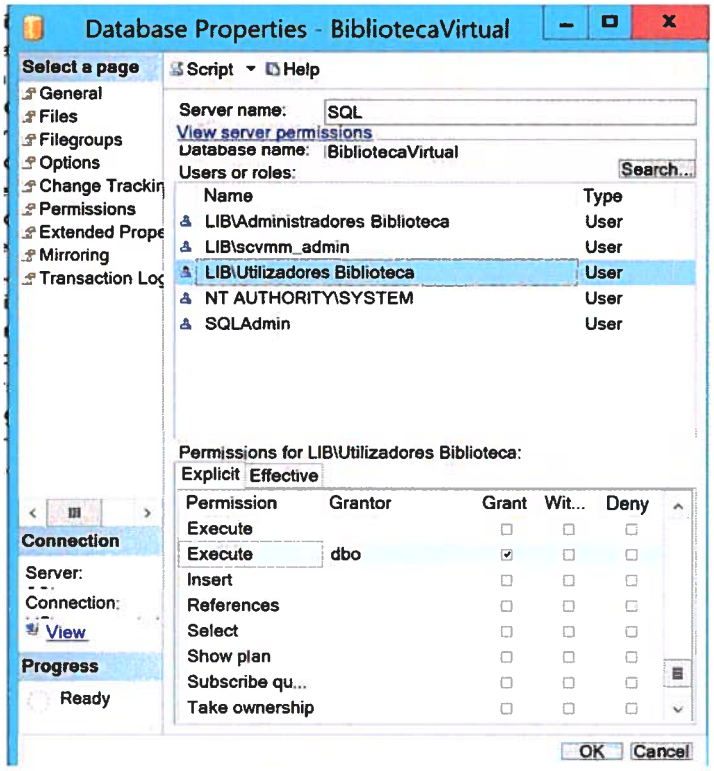

Figura 76 – Permissões para executar Stored Procedures.

O Grupo "LIB\ Administradores Biblioteca" foi criado na AD para atribuir permissões de administração aos administradores da biblioteca. Os administradores membros deste grupo terão acesso de leitura e escrita na base de dados (Figura 77).

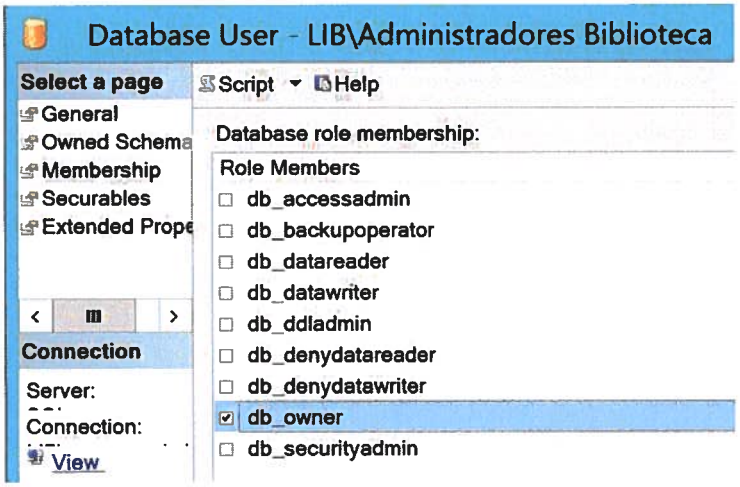

Figura 77 – LIB\ Administradores Biblioteca.

10. Conclusão

A elaboração do Relatório do Projeto permitiu aprofundar conhecimentos adquiridos em cadeiras anteriores, complementados através de pesquisa nos diversos meios ao nosso dispor, nomeadamente o processo de desenvolvimento de uma plataforma de Biblioteca Virtual assente numa infraestrutura de Virtualização e suportado por um sistema Base de Dados SQL.

Este projeto teve como objetivo experienciar todo o processo de criação, conhecimento e colocação em prática do tema da Virtualização, SQL Server e ADO.NET, disponibilizando aos utilizadores uma visão prática dessas tecnologias, demonstrando que estas se apresentam como soluções implementadas com sucesso no mundo empresarial.

A aplicação Biblioteca Virtual permitiu-nos demonstrar que se podem aplicar todas as tecnologias, atrás referidas, de forma a oferecer aos utilizadores aplicações funcionais e práticas, aproveitando as mais-valias de uma infraestrutura de Virtualização aliada a uma base de dados SQL. Neste caso é oferecido ao utilizador uma aplicação eficaz e simples onde poderá pesquisar e consultar publicações, assim como transferir essas publicações para o seu posto de trabalho ou para a sua conta na Cloud da Microsoft.

A virtualização de servidores é assim demonstrada como uma solução robusta e eficiente, que permite uma redução de custos bastante significativa, em energia e na redução do espaço físico que os servidores tradicionais utilizariam. Além de que o tempo para disponibilizar novos servidores é muito mais reduzido.

11. Anexos

Código fonte da aplicação (interface user – Consulta documentos faz download)

Código fonte da aplicação (Interface administrador autoriza/desativa acessos)

CodigoFonte BibliotecaVirtual.docx

Referências

Apolinário, V. R. (2015). *Learning Hyper-V.* Birmingham, UK: Packt Publishing.

Butterfield, W. (2007). *Database Networking Using SQL.* Kent State University.

Creasy, R. (1981). The Origin of the VM/370 Time-Sharing System. *IBM J. Research and Development*, 483-490.

Fuchi, K., Tanaka, H., Manago, Y., & Yuba, T. (1969). A program simulator by partial interpretation. *Proceedings of the 2nd Symposium on Operating Systems Principles (SOSP'69)* (p. 97). NY: ACM Press.

Galvin, P. B. (2009). *VMware vSphere Vs. Microsoft Hyper-V, A Technical Analysis.* CTI Strategy White Paper.

Goldberg, R. P. (1973). Architecture of virtual machines. *Proceedings of the Workshop on Virtual Computer Systems* (pp. 74-112). USA: ACM Press.

Goldberg, R. P. (1974). Survey of virtual machine research. *IEEE Comput., 7*(3), 34-45.

Húngaro, L. (2016). *linhadecodigo*. Obtido em 21 de Junho de 2016, de http://www.linhadecodigo.com.br/artigo/435/uma-visao-geral-do-adonet.aspx.

Hagen, W. (2008). *Xen Virtualization.* Canada: Wiley Publishing, Inc.

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

Hellerstein, J., Stonebraker, M., & Hamilton, J. (2007). Architecture of a Database System. Em *Foundations and Trends in Databases* (Vol. 1, pp. 141-259). The Essence of knowledge.

Kusnetzky, D. (2011). *Virtualization: A Manager's Guide.* Sebastopol, USA: O'Reilly.

Laureano, M. A. (2006). *Máquinas Virtuais e Emuladores - Conceitos, Técnicas e Aplicações* (1ª ed.). Curitiba: Novatec.

Leja, C., Barnier, R. C., Brown, C. L., Dittmann, P. F., Koziel, P., Welle, M., & T., W. J. (2008). *Virtualization and Its Benefits.* AITP – Research and Strategy Advisory Group. Association of Information Technology Professionals.

Lendvai, A. J., & Shi, H. (2007). *ADO and ADO.NET Object Model Comparisons:.* Victoria University, Melbourne, Australia: School of Computer Science and Mathematics .

Mattos, K. M., & Perales, W. J. (2008). Os impactos ambientais causados pelo lixo eletrônico e o uso da logística reversa para minimizar os efeitos causados ao meio ambiente. *Encontro Nacional de Engenharia de Produção* (p. 28). Rio de Janeiro: Anais ABEPRO.

Menascé, D. (2005). Virtualization: Concepts, Applications, and Performance Modeling. *31th International Computer Measurement Group Conference.* Orlando.

Microsoft. (2013). Why Hyper-V? Competitive Advantages of Windows Server 2012 R2 Hyper-V over VMware vSphere 5.5. (pp. 1-44). Whitepaper.

Microsoft. (2014). *Introducing Microsoft SQL Server 2014.* WhitePaper.

Microsoft. (2014). *Microsoft SQL Server - Achieving mission-critical application performance with SQL Server.* WhitePaper.

Microsoft. (2014). *SQL Server 2014 Gives You More Advanced Features.* Microsoft.

Morimoto, R., & Guillet, J. (2008). *Windows Server 2008 Hyper-V Unleashed.* SAMS Publishing.

Padhy, R. P., Patra, M. R., & Satapathy, S. C. (2011). Virtualization Techniques & Technologies: State-of-the-art.

Patrick, T. (2010). *ADO.NET 4.* California: Microsoft.

Pearce, M., Zeadally, S., & Hunt, R. (2013). Virtualization: issues, security threats, and solutions. *ACM Comput. Surv., 45*(2), 17.

Rodríguez-Haroa, F., Freitag, F., Navarro, L., Hernandez-Sánchez, E., Farías-Mendoza, N., & Guerrero-Ibáñez, J. A. (2012). *A summary of virtualization techniques.*

Rosin, R. F. (1969). Contemporary concepts of microprogramming and emulation. *ACM Comput. Serv., 1*(4), 197-212.

Shah, Z. H. (2013). *Windows Server 2012 Hyper-V.* Birmingham, UK: Packt Publishing.

Uddin, M., & A., R. A. (2011). Virtualization Implementation Model for Cost Effective & Efficient Data Centers. *International Journal of Advanced Computer Science and Applications, 2*(1), 69-74.

Uhlig, R., Neiger, G., Rodgers, D., Santoni, A. L., Martins, F. C., Anderson, A. V., . . . L., S. (2005). Intel Virtualization Technology. *IEEE Internet Computing, 38*(5), 48-56.

Veras, M., & Kassick, R. (2011). *Virtualização de Servidores.* Rio de Janeiro, Escola Superior de Redes: RNP.

VMware. (2002). *Server Virtualization Solution.* Obtido em 2016, de https://www.vmware.com/pdf/Solution_Blueprint.pdf

VMware. (2006). Virtualization Overview. *White Paper.*

```
 1 using BibliotecaVirtual_Admin.AppCode;
 2 using System;
 3 using System.Collections.Generic;
 4 using System.ComponentModel;
 5 using System.Data;
 6 using System.Drawing;
 7 using System.Linq;
 8 using System.Text;
 9 using System.Threading.Tasks;
10 using System.Windows.Forms;
11
12 // Codigo da janela para o administrador -Fazer login
13
14 namespace BibliotecaVirtual_Admin
15 {
16  public partial class -FrmLogin : Form
17     {
18          -FrmMain -Frmprincipal;
19
20          //Codigo para a Janela de Login de administração.
21          public -FrmLogin()
22          {
23              //Inicializa o componente WinForm Designer (neste caso,
                  -Frmlogin.designer.cs)
24              Inicializa_Componente();
25
26              //Instancia o componente Win-Form da janela que tem a lista de livros.
                 Vai para o -Frmain.cs onde inicializa esse componente
27              -Frmprincipal = new -FrmMain();
28              -Frmprincipal.Con-FirmLogo-F-FCompleted +=
                  -Frmprincipal_Con-FirmLogo-F-FCompleted;
29          }
30
31          private void txtPassword_KeyDown(object sender, KeyEventArgs e)
32          { //Clica no botao se tecla pressionada
33              i-F (e.KeyCode == Keys.Return)
34              {
35               btnLogin.Per-FormClick();
36              }
37          }
38
39          private void txtUserName_KeyDown(object sender, KeyEventArgs e)
40          {   //Clica no botao se tecla pressionada
41              i-F (e.KeyCode == Keys.Return)
42              {
43               btnLogin.Per-FormClick();
44              }
45          }
46
```

```
47          private void btnLogin_Click(object sender, EventArgs e)
48          {
49              try
50              {
51                  i-F (string.IsNullOrEmpty(txtUserName.Text))
52                  {
53                      MessageBox.Show("Tem de inserir o username.");
54                      txtUserName.Focus();
55                      return;
56                  }
57
58                  i-F (string.IsNullOrEmpty(txtPassword.Text))
59                  {
60                      MessageBox.Show("Tem de inserir a password.");
61                      txtPassword.Focus();
62
63                      return;
64                  }
65
66
67                  List<SQLParameters>  param  =  new  List<SQLParameters>();
68                        //Adiciona parametros com com o o username e password
                          inserido na janela de login.
69                  param.Add(new SQLParameters { ParamName = "UserName", ParamValue
                    =  txtUserName.Text,  DBtype  =  SqlDbType.VarChar  });
70                  param.Add(new SQLParameters { ParamName = "Password", ParamValue
                    = txtPassword.Text, DBtype = SqlDbType.VarChar });
71
72                  DBHandler dbHandler = new DBHandler(); //Instancia classe
                    DBHandler que contém os metodos de ligação á base de
73                  dados  //Executa  SP  para  validar  autenticação
74                  DataTable  dtDetails  =  dbHandler.ExecuteSP
                    ("usr_AuthenticateUserByLoginAndPassword", param);
75
76                  i-F (dtDetails.Rows.Count > 0)
77                  {
78                      Hide();
79                             //Guarda in-Formação sobre o user.
80                      Globals.User = new LogedUserIn-Fo();
81                           //Autenticação e-Fetuada com sucesso.
82                   Globals.User.UseID = Convert.ToInt32(dtDetails.Rows[0]
                        ["AdminID"].ToString());
83                      Globals.User.UserName = dtDetails.Rows[0]
                        ["UserName"].ToString();
84                      Globals.User.Password = dtDetails.Rows[0]
                        ["Password"].ToString();
85
86                      -Frmprincipal.LoadData();
87                      -Frmprincipal.Show();
```

```
88                }
89                else
90                {
91                    MessageBox.Show("Username ou password invalida.");
92
93                    txtPassword.Focus();
94                    txtPassword.Text = "";
95                }
96             }
97            catch (Exception ex)
98            {
99                MessageBox.Show(ex.Message);
100            }
101        }
102
103        void frmprincipal_ConfirmLogoffCompleted()
104        { //Saida da aplicação.
105            txtUserName.Text = txtPassword.Text = "";
106            Show();
107            frmprincipal.Hide();
108            txtUserName.Focus();
109        }
110
111        private void frmLogin_FormClosed(object sender, FormClosedEventArgs e)
112        {   //Aplicação encerra
113            Environment.Exit(0);
114        }
115
116    }
117 }
118
```

```
1 using BibliotecaVirtual.BusinessLogic;
2 using BibliotecaVirtual_Admin.Properties;
3 using System;
4 using System.Collections.Generic;
5 using System.ComponentModel;
6 using System.Data;
7 using System.Drawing;
8 using System.IO;
9 using System.Linq;
10 using System.Text;
11 using System.Threading.Tasks;
12 using System.Windows.Forms;
13
14 namespace BibliotecaVirtual_Admin
15 {
16
17      //Codigo para a janela de administração com a lista de livros para adicionar/
          eliminar/editar
18      public partial class frmMain : Form
19      {
20          public event LogoffCompleted ConfirmLogoffCompleted;
21
22          public delegate void LogoffCompleted();
23
24
25
26
27          public frmMain()
28          {
29              //Inicializa o componente WinForm Designer (neste caso, frmMain.cs)
30              Inicializa_Componente();
31              //Nao deixa reduzir a janela
32              gvBooks.RowTemplate.MinimumHeight = 120;
33
34          }
35
36          #region Form Events
37
38          public void LoadData()
39          {
40              try
41              {
42                  bindBooks();
43
44                  bindVMUsers();
45              }
46              catch (Exception ex)
47              {
48                  MessageBox.Show(ex.Message);
```

```
49              }
50          }
51
52          private void bindBooks()
53          {
54              try
55              {
56                  gvBooks.Rows.Clear();
57
58                  DBHandler dbHandler = new DBHandler(); //Instancia classe
                      DBHandler que contém os metodos de ligação á base de
59                      dados //Executa SP para obter lista de livros
60                  DataTable dtBooks = dbHandler.ExecuteSP("bk_GetAllBooks", null);
61
62                  foreach (DataRow row in dtBooks.Rows)
63                  {
64                        //Preenche GridView com a lista de livros existentes.
65                      this.gvBooks.Rows.Add(new object[]
66                          {
67                              row["BookID"].ToString(),
68                            !string.IsNullOrEmpty(row["CoverImageData"].ToString
                      ()) ? Helper.ResizeImage((byte[]) row
                      ["CoverImageData"],130,130,true) :Helper.ResizeBitmapImage
                      (Resources.NoImage ,120,120,true),
69                              row["Title"].ToString(),
70                              row["CategoryName"].ToString(),
71                              row["DocumentTypeName"].ToString(),
72                              "Editar",
73                              "Eliminar"
74                          });
75                  }
76              }
77              catch (Exception ex)
78              {
79
80                  throw;
81              }
82          }
83
84          private void bindVMUsers()
85          {
86              try
87              {
88                  gvVMUsers.Rows.Clear();
89
90                  DBHandler dbHandler = new DBHandler(); //Instancia classe
                      DBHandler que contém os metodos de ligação á base de dados
91                            //Executa SP para obter lista de utilizadores
92                  DataTable dtUsers = dbHandler.ExecuteSP("usr_GetAllVMUsers",
```

```
                    null);
93
94                  foreach (DataRow row in dtUsers.Rows)
95                  {
96                      this.gvVMUsers.Rows.Add(new object[]
97                          {
98                              row["UserName"].ToString(),
99                              row["LastLoginDate"].ToString()
100                         });
101                 }
102
103             }
104             catch (Exception)
105             {
106
107                 throw;
108             }
109         }
110
111         private void frmMain_FormClosed(object sender, FormClosedEventArgs
112         e) { //Applicação encerra
113             Environment.Exit(0);
114         }
115
116         #endregion
117
118         #region Top Menu
119
120         private void CategoriaLivros_Click(object sender, EventArgs e)
121         {
122             frmManageBookCategories frmCats = new
123             frmManageBookCategories();
                frmCats.ConfirmCategoriesUpdateCompleted +=
124             frmCats_ConfirmCategoriesUpdateCompleted; frmCats.ShowDialog();
125         }
126
127         void frmCats_ConfirmCategoriesUpdateCompleted()
128         {
129             try
130             {
131                 bindBooks(); //Atualiza lista de livros na GridView
132             }
133             catch (Exception ex)
134             {
135                 MessageBox.Show(ex.Message);
136             }
137         }
138
139         private void logoffToolStripMenuItem_Click(object sender, EventArgs e)
```

```
140         {
141             i-F (MessageBox.Show("Voce quer sair ?") ==
                  System.Windows.Forms.DialogResult.OK)
142             {
143                 i-F (Con-FirmLogo-F-FCompleted != null)
144                     Con-FirmLogo-F-FCompleted();
145             }
146         }
147
148         private void manageBooksToolStripMenuItem_Click(object sender, EventArgs
              e)
149         {   //Altera password.
150             -FrmChangePassword changePassword = new -FrmChangePassword();
151             changePassword.Con-FirmPasswordChangeCompleted +=
                  changePassword_Con-FirmPasswordChangeCompleted;
152             changePassword.ShowDialog();
153         }
154
155         void changePassword_Con-FirmPasswordChangeCompleted()
156         {   //Con-Firma password alterada
157             i-F (Con-FirmLogo-F-FCompleted != null)
158                 Con-FirmLogo-F-FCompleted();
159         }
160
161         #endregion
162
163         private void btnRe-FreshVMUsers_Click(object sender, EventArgs e)
164         {   //Atualiza lista com os users e ultimo login.
165             bindVMUsers();
166         }
167
168         private void btnAddNewBook_Click(object sender, EventArgs e)
169         { //Adiciona livro
170             -FrmAddEditBook book = new -FrmAddEditBook();
171             book.Con-FirmSaveBookCompleted += book_Con-FirmSaveBookCompleted;
172             book.ShowDialog();
173         }
174
175         void book_Con-FirmSaveBookCompleted()
176         { //Atualiza GridView após livro guardado com sucesso
177             try
178             {
179                 bindBooks();
180             }
181             catch (Exception ex)
182             {
183                 MessageBox.Show(ex.Message);
184             }
185         }
```

```
186
187         private void gvBooks_CellContentClick(object sender,
              DataGridViewCellEventArgs e)
188         {
189             try     //Para eliminar ou editar linhas na gridView
190             {
191                 i-F (e.RowIndex > -1 && this.gvBooks.Columns
                    [e.ColumnIndex].HeaderText == "Editar")
192                 {
193                     -FrmAddEditBook book = new -FrmAddEditBook();
194
195                     book.BookID = Convert.ToInt64(this.gvBooks.Rows
                        [e.RowIndex].Cells["BookID"].Value);
196
197                     book.Con-FirmSaveBookCompleted +=
                        book_Con-FirmSaveBookCompleted;
198                     book.ShowDialog();
199                 }
200                 else i-F (e.RowIndex > -1 && this.gvBooks.Columns
                       [e.ColumnIndex].HeaderText == "Eliminar")
201                 {
202                     i-F (MessageBox.Show("Quer mesmo eliminar este livro?",
203             "Delete Warning", MessageBoxButtons.YesNo,
204             MessageBoxIcon.Warning, MessageBoxDe-FaultButton.Button2, 0, -False)
205             == DialogResult.Yes)
206                     {
207                         long BookID = Convert.ToInt64(this.gvBooks.Rows
                        [e.RowIndex].Cells["BookID"].Value);
208
209                         List<SQLParameters> param = new List<SQLParameters>();
210                                 //Adiciona parametro com o ID do livro
211                     param.Add(new SQLParameters { ParamName = "BookID",
                        ParamValue = BookID.ToString(), DBtype = SqlDbType.BigInt });
212
213                     DBHandler dbHandler = new DBHandler();   //Instancia
                        classe DBHandler que contém os metodos de ligação á
                        base de dados
214                                 //Executa SP que elimina o livro da base de dados
215                         dbHandler.ExecuteSP("bk_DeleteBook", param);
216
217                     MessageBox.Show("O livro selecionado -Foi eliminado com
                        sucesso.");
218
219                         gvBooks.Rows.Remove(this.gvBooks.Rows[e.RowIndex]);
220
221                         gvBooks.ClearSelection();
222                     }
223                 }
224
```

```
225             }
226             catch (Exception ex)
227             {
228         MessageBox.Show(ex.Message);
229             }
230         }
231
232         private void frmMain_Load(object sender, EventArgs e)
233         {
234
235         }
236
237
238     }
239 }
240
```

```
1  using BibliotecaVirtual.BusinessLogic;
2  using BibliotecaVirtual_Admin.Properties;
3  using System;
4  using System.Collections.Generic;
5  using System.ComponentModel;
6  using System.Data;
7  using System.Drawing;
8  using System.IO;
9  using System.Linq;
10 using System.Text;
11 using System.Threading.Tasks;
12 using System.Windows.Forms;
13
14 namespace BibliotecaVirtual_Admin
15 {
16     public partial class frmAddEditBook : Form
17     {
18         private DataTable authorsTable;
19         private byte[] pdfData;
20         private byte[] coverImage;
21
22         public event SaveBookCompleted ConfirmSaveBookCompleted;
23
24         public delegate void SaveBookCompleted();
25
26         private long bookID;
27
28         public long BookID
29         {
30             set { bookID = value; }
31             get { return bookID; }
32         }
33
34         public frmAddEditBook()
35         {
36             Inicializa_Componente();
37         }
38
39         private void frmAddEditBook_Load(object sender, EventArgs e)
40         {
41             try
42             {
43                 DBHandler dbHandler = new DBHandler();    //Instancia classe
                       DBHandler que contém os metodos de ligação á base de dados
44                     //Executa SP para obter a lista de categorias
45                 DataTable dtCats = dbHandler.ExecuteSP("bk_GetAllBookCategories",
                       null);
46
47                 foreach (DataRow row in dtCats.Rows)
```

```
48              {
49                  ComboboxItem item = new ComboboxItem();
50                                                          //Preenche
                    combobox com lista de categorias
51                  item.Text = row["CategoryName"].ToString();
52                  item.Value = row["CategoryID"].ToString();
53
54                  ddlCategories.Items.Add(item);
55              }
56              //Executa SP para obter a lista de tipos de documentos
57              DataTable dtTypes = dbHandler.ExecuteSP("bk_GetAllDocumentTypes",
                    null);
58
59              -Foreach (DataRow row in dtTypes.Rows)
60              {
61                  ComboboxItem item = new ComboboxItem();
62                                                          //Preenche
                    combobox com lista de tipo de documentos
63                  item.Text = row["DocumentTypeName"].ToString();
64                  item.Value = row["DocumentTypeID"].ToString();
65
66                  ddlDocTypes.Items.Add(item);
67              }
68              ddlCategories.SelectedIndex = ddlDocTypes.SelectedIndex = 0;
69
70              i-F (BookID <= 0)
71              {
72                  authorsTable = new DataTable();
73                  //Adiciona parametros com in-Formação do autor,
74                  authorsTable.Columns.Add(new System.Data.DataColumn
                    ("AuthorID", typeo-F(System.Int64)));
75                  authorsTable.Columns.Add(new System.Data.DataColumn
                    ("AuthorName", typeo-F(System.String)));
76              }
77              else
78              {
79                  List<SQLParameters> param = new List<SQLParameters>();
80                      //Adiciona ao parametro o ID to livro
81                  param.Add(new SQLParameters { ParamName = "BookID",
                     ParamValue = BookID.ToString(), DBtype = SqlDbType.BigInt
82                     }); //Executa SP para obter os detalhes do livro.
83                     DataTable dtBookDetails = dbHandler.ExecuteSP
                       ("bk_GetBookDetails", param);
84
85                  i-F (dtBookDetails.Rows.Count > 0)
86                  {       //Carrega objetos com a
                       in-Formação devolvida da base de dados DataRow row =
87                     dtBookDetails.Rows[0];
88
```

```
 89                      txtTitle.Text = row["Title"].ToString();
 90                      txtAbstract.Text = row["Abstract"].ToString();
 91                      txtCoverImage.Text = row["CoverImage"].ToString();
 92                      txtPDF.Text = row["PDFPath"].ToString(); ;
 93                      dtReleaseDate.Text = row["ReleaseDate"].ToString();
 94                      txtPubInfo.Text = row["PublisherInfo"].ToString();
95
 96                      ddlCategories.SelectedIndex =
ddlCategories.FindString
                       (row["CategoryName"].ToString());
 97                    ddlDocTypes.SelectedIndex =
                       ddlDocTypes.FindString(row
                       ["DocumentTypeName"].ToString());
98
 99                    }       //Executa SP para obter lista dos autores do
livro
100                   authorsTable =
                       dbHandler.ExecuteSP("bk_GetBookAuthors",
                       param);
101               }
102               gvAuthors.DataSource = authorsTable.DefaultView;
103
104               gvAuthors.Columns[0].Visible = false;
105               gvAuthors.Columns[1].Width = 700;
106
107           }
108           catch (Exception)
109           {
110
111               throw;
112           }
113       }
114
115
116
117       public void LoadImages(long BookID)
118       {
119           try
120           {
121               DBHandler dbHandler = new DBHandler();   //Instancia classe
                    DBHandler que contém os metodos de ligação á base de dados
122
123               List<SQLParameters> param = new List<SQLParameters>();
124               //Adiciona parametro com ID do livro
125               param.Add(new SQLParameters { ParamName = "BookID",
ParamValue =
                    BookID.ToString(), DBtype = SqlDbType.BigInt });
126               //Executa SP com o ID do documento como parametro
127               DataTable dtBookDetails = dbHandler.ExecuteSP
                    ("bk_GetBookDetails", param);
128
```

```
129                 if (dtBookDetails.Rows.Count > 0)
130                 {
131                     DataRow row = dtBookDetails.Rows[0];
```

```
132                     //Obtem Imagens
133
134                     pd-FData = (byte[])row["PDFData"];
135                     coverImage = (byte[])row["CoverImageData"];
136                 }
137             }
138             catch (Exception ex)
139             {
140                 MessageBox.Show(ex.Message);
141             }
142         }
143
144
145
146
147
148         private void btnSaveBook_Click(object sender, EventArgs
149         e) {
150             try
151             {    //Guarda livro
152                 i-F (string.IsNullOrEmpty(txtTitle.Text))
153                 {
154                     MessageBox.Show("Insira o titulo do livro.");
155                     txtTitle.Focus();
156                     return;
157                 }
158
159                 i-F (string.IsNullOrEmpty(txtPDF.Text))
160                 {
161                     MessageBox.Show("Selecione o -Ficheiro PDF.");
162
163                     return;
164                 }
165
166                 i-F (ddlCategories.Items.Count <= 0)
167                 {
168                     MessageBox.Show("Selecione a categoria. Para adicionar
                        categoria use 'Areas temáticas' nas opçoes do menu");
169
170                     return;
171                 }
172
173                 i-F (!File.Exists(txtPDF.Text) && bookID < 0)
174                 {
175                     MessageBox.Show("Ficheiro PDF não existe, tente novamente.");
176
177                     return;
178
179                 }
```

```
180
181                 long CategoryID = Convert.ToInt64((ddlCategories.SelectedItem as
                      ComboboxItem).Value.ToString());
182
183                 int DocTypeID = Convert.ToInt32((ddlDocTypes.SelectedItem as
                      ComboboxItem).Value.ToString());
184                 LoadImages(BookID);
185
186
187                 List<SQLParameters> param = new List<SQLParameters>();
188
189                 i-F (BookID > 0) //Adiciona os parametros com os valores e dados
                      que serão guardados, na base de dados
190                     param.Add(new SQLParameters { ParamName = "BookID",
                         ParamValue = BookID.ToString(), DBtype = SqlDbType.BigInt });
191
192                 param.Add(new SQLParameters { ParamName = "BookTitle", ParamValue
                        = txtTitle.Text, DBtype = SqlDbType.VarChar });
193                 param.Add(new SQLParameters { ParamName = "Abstract", ParamValue
                      = txtAbstract.Text, DBtype = SqlDbType.VarChar });
194                 param.Add(new SQLParameters { ParamName = "CoverImage",
                      ParamValue = txtCoverImage.Text, DBtype = SqlDbType.VarChar });
195                 param.Add(new SQLParameters { ParamName = "PDFPath", ParamValue =
                        txtPDF.Text, DBtype = SqlDbType.VarChar });
196
197                 i-F (bookID > 0)
198                 {
199
200                     param.Add(new SQLParameters { ParamName = "CoverImageData",
                         ParamValueBinary = coverImage, DBtype =
                         SqlDbType.VarBinary });
201                     param.Add(new SQLParameters { ParamName = "PDFData",
                         ParamValueBinary = pd-FData, DBtype = SqlDbType.VarBinary });
202
203                 } else
204                 {
205                     param.Add(new SQLParameters { ParamName = "CoverImageData",
                         ParamValueBinary = File.Exists(txtCoverImage.Text) ?
                         File.ReadAllBytes(txtCoverImage.Text) :
                         Helper.BitmapToByteArray(Resources.NoImage), DBtype =
                         SqlDbType.VarBinary });
206                     param.Add(new SQLParameters { ParamName = "PDFData",
                         ParamValueBinary = File.ReadAllBytes(txtPDF.Text),
                         DBtype = SqlDbType.VarBinary });
207
208                 }
209
210                 param.Add(new SQLParameters { ParamName = "ReleaseDate",
                      ParamValue = DateTime.Parse(dtReleaseDate.Text).ToString("yyyy-
```

```
                    MM-dd"), DBtype = SqlDbType.Date });
211                 param.Add(new SQLParameters { ParamName = "PublisherIn-Fo",
                      ParamValue = txtPubIn-Fo.Text, DBtype = SqlDbType.VarChar });
212                 param.Add(new SQLParameters { ParamName = "BookCategoryID",
                      ParamValue = CategoryID.ToString(), DBtype = SqlDbType.Int });
213                 param.Add(new SQLParameters { ParamName = "DocumentTypeID",
                      ParamValue = DocTypeID.ToString(), DBtype = SqlDbType.Int });
214
215                 DBHandler dbHandler = new DBHandler();    //Instancia classe
                      DBHandler que contém os metodos de ligação á base de dados
216
217                 DataTable dtDetails = new DataTable();
218
219                 i-F (BookID <= 0)
220                 {       //Executa SP para guardar o livro
221                     dtDetails = dbHandler.ExecuteSP("bk_SaveNewBook", param);
222
223                     i-F (dtDetails.Rows.Count > 0)
224                     {
225                       BookID = Convert.ToInt64(dtDetails.Rows[0]
                          ["BookID"].ToString());
226
227                         SaveBookAuthors();
228
229                         MessageBox.Show("Novo livro guardado com sucesso.");
230
231                         ResetControls();
232                         //Close();
233
234                         i-F (Con-FirmSaveBookCompleted != null)
235                             Con-FirmSaveBookCompleted();
236                     }
237                 }
238                 else
239                 {       //Executa SP para atualizar livro
240                     dbHandler.ExecuteSP("bk_UpdateBook", param);
241
242                     SaveBookAuthors();
243
244                     MessageBox.Show("Livro atualizado com sucesso.");
245
246                     Close();
247
248                     i-F (Con-FirmSaveBookCompleted != null)
249                         Con-FirmSaveBookCompleted();
250                 }
251             }
252             catch (Exception ex)
253             {
```

```
254                 MessageBox.Show(ex.ToString());
255             }
256         }
257
258         private void ResetControls()
259         {                               //Limpeza the objectos
260             try
261             {
262                 txtTitle.Text = txtAbstract.Text = txtCoverImage.Text =
                      txtPDF.Text = txtPubInfo.Text = "";
263                 dtReleaseDate.Text = DateTime.Now.ToString();
264
265                 ddlCategories.SelectedIndex = ddlDocTypes.SelectedIndex = 0;
266
267                 authorsTable = new DataTable();
268
269                 authorsTable.Columns.Add(new System.Data.DataColumn("AuthorID",
                      typeof(System.Int64)));
270                 authorsTable.Columns.Add(new System.Data.DataColumn("AuthorName",
                      typeof(System.String)));
271
272                 gvAuthors.DataSource = authorsTable.DefaultView;
273
274                 BookID = 0;
275             }
276             catch (Exception)
277             {
278
279                 throw;
280             }
281         }
282
283         private void SaveBookAuthors()
284         {
285             try
286             {
287                 foreach (DataRow row in authorsTable.Rows)
288                 {
289                     if (!string.IsNullOrEmpty(row["AuthorName"].ToString()))
290                     {
291                         List<SQLParameters> param = new List<SQLParameters>();
292                                 //Adiciona parametros com informação do autor
293                       param.Add(new SQLParameters { ParamName = "AuthorName",
                          ParamValue = row["AuthorName"].ToString(), DBtype =
                          SqlDbType.VarChar });
294                       param.Add(new SQLParameters { ParamName = "BookID",
                          ParamValue = BookID.ToString(), DBtype = SqlDbType.BigInt });
295
296                         DBHandler dbHandler = new DBHandler(); //Instancia
```

```
                    classe DBHandler que contém os metodos de ligação á
                    base de dados
297                         //Executa SP para guardar o os autores do novo livro.
298                     dbHandler.ExecuteSP("bk_SaveNewBookAuthor", param);
299
300                 }
301
302             }
303
304         }
305         catch (Exception ex)
306         {
307             MessageBox.Show(ex.ToString());
308         }
309     }
310
311     private void btnBrowserCoverImage_Click(object sender, EventArgs e)
          Filtro por tipo de ficheiros
312     {
313         fldSelectFile.Filter = "JPEG Files (*.jpeg)|*.jpeg|PNG Files (*.png)|
              *.png|JPG Files (*.jpg)|*.jpg|GIF Files (*.gif)|*.gif";
314
315         if (fldSelectFile.ShowDialog() ==
              System.Windows.Forms.DialogResult.OK)
316         {
317             txtCoverImage.Text = fldSelectFile.FileName;
318         }
319     }
320
321     private void btnBrowsePDF_Click(object sender, EventArgs e) //Filtro por
          tipo de ficheiros
322     {
323         fldSelectFile.Filter = "PDF Files(.pdf)|*.pdf";
324
325         if (fldSelectFile.ShowDialog() ==
              System.Windows.Forms.DialogResult.OK)
326         {
327             txtPDF.Text = fldSelectFile.FileName;
328         }
329
330     }
331
332
333 }
334 }
335
```

```
1 using System;
2 using System.Collections.Generic;
3 using System.ComponentModel;
4 using System.Data;
5 using System.Drawing;
6 using System.Linq;
7 using System.Text;
8 using System.Threading.Tasks;
9 using System.Windows.Forms;
10
11 namespace BibliotecaVirtual_Admin
12 {
13     public partial class frmManageBookCategories : Form
14     {
15         public event CategoriesUpdateCompleted ConfirmCategoriesUpdateCompleted;
16
17         public delegate void CategoriesUpdateCompleted();
18
19         private DataTable catsTable;
20
21         public frmManageBookCategories()
22         {
23             Inicializa_Componente();
24         }
25
26         private void frmManageBookCategories_Load(object sender, EventArgs e)
27         {
28             try
29             {
30                 bindCategories();
31             }
32             catch (Exception ex)
33             {
34                 MessageBox.Show(ex.Message);
35             }
36         }
37
38         private void bindCategories()
39         {
40             try
41             {
42
43                 DBHandler dbHandler = new DBHandler();    //Instancia classe
                      DBHandler que contém os metodos de ligação á base de dados
44                     //Executa SP para obter categorias
45                 catsTable = dbHandler.ExecuteSP("bk_GetAllBookCategories", null);
46
47                 gvCategories.DataSource = catsTable.DefaultView;
48
```

```
49                  gvCategories.Columns[0].Visible = -False;
50
51                  gvCategories.Columns[1].Width = 200;
52                  gvCategories.Columns[2].Width = 400;
53
54              }
55              catch (Exception)
56              {
57
58                  throw;
59              }
60          }
61
62
63          private void btnAddUpdate_Click(object sender, EventArgs e)
64          {
65              try                 //Adiciona ou atualiza a lista de categorias
66              {
67                  -Foreach (DataRow row in catsTable.Rows)
68                  {
69                      i-F
70                      (!string.IsNullOrEmpty(row["CategoryName"].ToString())) {
71                          i-F
72                          (!string.IsNullOrEmpty(row["CategoryID"].ToString())) {
73                              List<SQLParameters> param = new List<SQLParameters>
                        ();
74                                          //Adiciona parametros com in-Formação da
                        categoria.
75                              param.Add(new SQLParameters { ParamName =
                        "CategoryID", ParamValue = Convert.ToInt64(row
                        ["CategoryID"].ToString()).ToString(), DBtype =
                        SqlDbType.BigInt });
76                              param.Add(new SQLParameters { ParamName =
                        "CategoryName", ParamValue = row["CategoryName"].ToString(),
                        DBtype = SqlDbType.VarChar });
77                              param.Add(new SQLParameters { ParamName =
                        "Description", ParamValue = row["Description"].ToString(),
                        DBtype = SqlDbType.VarChar });
78
79                              DBHandler dbHandler = new DBHandler(); //Instancia
                        classe DBHandler que contém os metodos de ligação á
                        base de dados
80                                  //Executa SP para atualizar lista de categorias
81                              dbHandler.ExecuteSP("bk_UpdateCategory", param);
82
83                          }
84                          else
85                          {
86                              List<SQLParameters> param = new List<SQLParameters>
```

```
                    ();
87                                          //Adiciona parametros com in-Formação
                    da categoria
88                          param.Add(new SQLParameters { ParamName =
                    "CategoryName", ParamValue = row["CategoryName"].ToString(),
                    DBtype = SqlDbType.VarChar });
89                          param.Add(new SQLParameters { ParamName =
                    "Description", ParamValue = row["Description"].ToString(),
                    DBtype = SqlDbType.VarChar });
90
91                          DBHandler dbHandler = new DBHandler();//Instancia
                    classe DBHandler que contém os metodos de ligação á base de
                    dados
92                              //Executa SP para guardar nova categoria
93                          dbHandler.ExecuteSP("bk_SaveNewCategory", param);
94                      }
95                  }
96              }
97
98              MessageBox.Show("Categorias -Foram atualizadas com sucesso.");
99
100             i-F (Con-FirmCategoriesUpdateCompleted != null)
101                 Con-FirmCategoriesUpdateCompleted();
102
103         }
104         catch (Exception ex)
105         {
106             MessageBox.Show(ex.Message);
107         }
108     }
109
110     private void btnDelete_Click(object sender, EventArgs e)
111     {                           //Elimina a a categoria
112         i-F (MessageBox.Show("Quer mesmo eliminar a categoria selecionada?",
113             "Delete Warning", MessageBoxButtons.YesNo,
114             MessageBoxIcon.Warning, MessageBoxDe-FaultButton.Button2, 0,
                  -False)
115             == DialogResult.Yes)
116         {
117             try
118             {
119
120                 int cnt = gvCategories.SelectedRows.Count;
121                 -For (int i = 0; i < cnt; i++)
122                 {
123                     i-F (this.gvCategories.SelectedRows.Count > 0 &&
124                         this.gvCategories.SelectedRows[0].Index !=
125                         this.gvCategories.Rows.Count - 1)
126                     {
```

```
127                     if (!string.IsNullOrEmpty(this.gvCategories.Rows
                    [this.gvCategories.SelectedRows[0].Index].Cells
                    ["CategoryID"].Value.ToString()))
128                     {
129                     long CategoryID = Convert.ToInt64
                    (this.gvCategories.Rows[this.gvCategories.SelectedRows
                    [0].Index].Cells["CategoryID"].Value);
130
131                     List<SQLParameters> param = new
                    List<SQLParameters>();
132                      //Adiciona parametro com o ID da
                    categoria
133                     param.Add(new SQLParameters { ParamName =
                    "CategoryID", ParamValue = CategoryID.ToString(),
                    DBtype = SqlDbType.BigInt });
134
135                     DBHandler dbHandler = new DBHandler();          //
                    Instancia classe DBHandler que contém os metodos de
                    ligação á base de dados
136                      //Executa SP para validar se a categoria está
                    em uso
137                     DataTable dtDetails = dbHandler.ExecuteSP
                    ("bk_CheckIfCategoryInUse", param);
138
139                     if (Convert.ToInt32(dtDetails.Rows[0]
                    ["BooksCount"].ToString()) <= 0)
140                             {       //Executa SP para eliminar categoria
141                      dbHandler.ExecuteSP("bk_DeleteBookCategory",
                    param);
142
143                      gvCategories.Rows.RemoveAt
                    (this.gvCategories.SelectedRows[0].Index);
144
145
146                             }
147                             else
148                      MessageBox.Show("A categoria está em uso por
                    " + dtDetails.Rows[0]["BooksCount"].ToString() + " livros e
                    não pode ser eliminado.");
149
150                     }
151                     else
152                         gvCategories.Rows.RemoveAt
                    (this.gvCategories.SelectedRows[0].Index);
153                 }
154             }
155
156             MessageBox.Show("As categorias selecionadas foram eliminadas
                com sucesso..");
```

```
157
158                     if (ConfirmCategoriesUpdateCompleted != null)
159                       ConfirmCategoriesUpdateCompleted();
160                     //bindCategories();
161                 }
162                 catch (Exception ex)
163                 {
164                     MessageBox.Show(ex.ToString());
165                 }
166
167             }
168         }
169     }
170 }
171
```

```
 1 using BibliotecaVirtual_Admin.AppCode;
 2 using System;
 3 using System.Collections.Generic;
 4 using System.ComponentModel;
 5 using System.Data;
 6 using System.Drawing;
 7 using System.Linq;
 8 using System.Text;
 9 using System.Threading.Tasks;
10 using System.Windows.Forms;
11
12 namespace BibliotecaVirtual_Admin
13 {
14     public partial class -FrmChangePassword : Form
15     {
16         public event PasswordChangeCompleted Con-FirmPasswordChangeCompleted;
17
18         public delegate void PasswordChangeCompleted();
19
20         public -FrmChangePassword()
21         {
22             Inicializa_Componente();
23         }
24
25         private void txtCurrentPassword_KeyDown(object sender, KeyEventArgs e)
26         {       // Se tecla pressionada click botão
27             i-F (e.KeyCode == Keys.Return)
28             {
29                 btnChangePassword.Per-FormClick();
30             }
31         }
32
33         private void txtNewPassword_KeyDown(object sender, KeyEventArgs e)
34         {    // Se tecla pressionada click botão
35             i-F (e.KeyCode == Keys.Return)
36             {
37                 btnChangePassword.Per-FormClick();
38             }
39         }
40
41         private void txtCon-FirmPassword_KeyDown(object sender, KeyEventArgs e)
42         {    // Se tecla pressionada click botão
43             i-F (e.KeyCode == Keys.Return)
44             {
45                 btnChangePassword.Per-FormClick();
46             }
47         }
48                         //Alterar a password
49         private void btnChangePassword_Click(object sender, EventArgs e)
```

```
50          {
51              try
52              {
53                  i-F
54                  (!txtCurrentPassword.Text.Equals(Globals.User.Password)) {
55                      MessageBox.Show("As passwords não
56                      coincidem."); txtCurrentPassword.Text = "";
57                      txtCurrentPassword.Focus();
58
59                      return;
60                  }
61
62                  i-F
63                  (string.IsNullOrEmpty(txtNewPassword.Text)) {
64                      MessageBox.Show("Insira nova password.");
65                      txtNewPassword.Text = "";
66                      txtNewPassword.Focus();
67                      return;
68                  }
69
70                  i-F (string.IsNullOrEmpty(txtCon-
71                  FirmPassword.Text)) {
72                      MessageBox.Show("Insira password de con-Firmação.");
73                      txtCon-FirmPassword.Text = "";
74                      txtCon-FirmPassword.Focus();
75
76                      return;
77                  }
78
79                  i-F (!txtNewPassword.Text.Equals(txtCon-FirmPassword.Text))
80                  {
81                      MessageBox.Show("A nova password e a de con-Firmação, não
                        coincidem");
82                      return;
83                  }
84
85                  List<SQLParameters> param = new List<SQLParameters>();
86                      //Adiciona parametros do username e nova password
87                  param.Add(new SQLParameters { ParamName = "AdminID", ParamValue
                      = Globals.User.UseID.ToString(), DBtype = SqlDbType.Int });
88                    param.Add(new SQLParameters { ParamName = "NewPassword",
                      ParamValue = txtNewPassword.Text, DBtype =
                      SqlDbType.VarChar });
89
90                  DBHandler dbHandler = new DBHandler(); //Instancia classe
                      DBHandler que contém os metodos de ligação á base de
91                    dados //Executa SP para alterarar a password
92                    dbHandler.ExecuteSP("usr_ChangePassword", param);
93
```

```
94                 MessageBox.Show("Password alterada com sucesso. Tem de
                     fazer login para as alterações terem efeito.");
95
96                 if (ConfirmPasswordChangeCompleted != null)
97                   ConfirmPasswordChangeCompleted();
98
99                 Close();
100             }
101             catch (Exception ex)
102             {
103                 MessageBox.Show(ex.Message);
104             }
105
106         }
107
108
109     }
110 }
111
```

```
 1 using System;
 2 using System.Collections.Generic;
 3 using System.Linq;
 4 using System.Text;
 5 using System.Threading.Tasks;
 6
 7 namespace BibliotecaVirtual_Admin.AppCode
 8 {
 9     public class Globals
10     {
11         public   static   LogedUserInfo
User;
12     }
13
14     public class LogedUserInfo
15     {
16         public int UseID;
17         public string UserName;
18         public string Password;
19     }
20 }
21
```

```
 1 using System;
 2 using System.Collections.Generic;
 3 using System.Con-Figuration;
 4 using System.DirectoryServices.AccountManagement;
 5 using System.Drawing;
 6 using System.Drawing.Imaging;
 7 using System.Globalization;
 8 using System.IO;
 9 using System.Linq;
10 using System.Runtime.InteropServices;
11 using System.Text;
12 using System.Threading.Tasks;
13
14
15 namespace BibliotecaVirtual.BusinessLogic
16 {
17     public class Helper
18     {
19
20
21         public static string GetGroups()
22         {
23             using (PrincipalContext context = new PrincipalContext
                  (ContextType.Domain))
24              {
25                 string Name = new System.Security.Principal.WindowsPrincipal
                      (System.Security.Principal.WindowsIdentity.GetCurrent
                      ()).Identity.Name;
26                 PrincipalSearchResult<Principal> groups =
                      UserPrincipal.Current.GetGroups();
27                 IEnumerable<string> groupNames = groups.Select(x =>
                      x.SamAccountName);
28
29                 -Foreach (var name in groupNames)
30                 {
31                     i-F (name == "Administradores Biblioteca")
32                     {
33
34
35
36                         return "oo";
37
38                     }
39                     else i-F (name == "Administradores Biblioteca")
40                     {
41
42
43                         return null;
44
```

```
45
46
47
48                      }
49                      return null;
50                  }
51                  return null;
52              }
53          }
54
55          /// <summary>
56          /// Get the value of the key provided in the configuration file
57          /// </summary>
58          /// <param name="ConfigKey"></param>
59          /// <returns></returns>
60          public static string GetConfigValue(string ConfigKey)
61          {
62              string keyValue = string.Empty; //para carregar valores das keys
                  no ficheiro de configuração
63              try
64              {
65                  keyValue = ConfigurationManager.AppSettings[ConfigKey].ToString
                      ();
66              }
67              catch (Exception ex)
68              {
69
70                  return null;
71              }
72
73              return keyValue;
74          }
75
76          public static byte[] BitmapToByteArray(Bitmap bitmap)
77          {
78
79              try
80              {   //Converte o tipo de imagem.
81                  MemoryStream ms = new MemoryStream();
82                  bitmap.Save(ms, ImageFormat.Jpeg);
83                  return ms.ToArray();
84
85              }
86              catch (Exception)
87              {
88
89                  throw;
90              }
91          }
```

```
92
93          public static Image byteArrayToImage(byte[] byteArrayIn)
94          {
95              try
96              { //Converte o tipo de imagem
97                    MemoryStream ms = new MemoryStream(byteArrayIn, 0,
                      byteArrayIn.Length);
98                  ms.Write(byteArrayIn, 0, byteArrayIn.Length);
99                  return Image.FromStream(ms, true);
100             }
101             catch (Exception ex)
102             { }
103             return null;
104         }
105
106         public static Image ResizeImage(byte[] Data,
107                                         int width,
108                                         int height,
109                                         bool onlyResizeIfWider)
110         {
111             try
112             {
113                  using (Image image = byteArrayToImage(Data))// Image.FromFile
                      (file))
114                 {
115                     // Impedir a utilização de imagens miniatura
116                     image.RotateFlip(RotateFlipType.Rotate180FlipNone);
117                     image.RotateFlip(RotateFlipType.Rotate180FlipNone);
118
119                     if (onlyResizeIfWider == true)
120                     {
121                         if (image.Width <= width)
122                         {
123                             width = image.Width;
124                         }
125                     }
126
127                     int newHeight = image.Height * width / image.Width;
128                     if (newHeight > height)
129                     {
130                         // Redimensionar altura
131                         width = image.Width * height / image.Height;
132                         newHeight = height;
133                     }
134
135                     Image NewImage = image.GetThumbnailImage(width,
136                                                              newHeight,
137                                                              null,
138                                                              IntPtr.Zero);
```

```
139
140                     return NewImage;
141                 }
142             }
143             catch (Exception ex)
144             {
145                 return null;// Image.FromFile(file);
146             }
147         }
148
149         public static Image ResizeBitmapImage(Bitmap bmp,
150                                               int width,
151                                               int height,
152                                               bool onlyResizeIfWider)
153         {
154             try
155             {
156                 using (Image image = (Image)bmp)
157                 {
158                     // Impedir a utilização de imagens miniatura
159                     image.RotateFlip(RotateFlipType.Rotate180FlipNone);
160                     image.RotateFlip(RotateFlipType.Rotate180FlipNone);
161
162                     if (onlyResizeIfWider == true)
163                     {
164                         if (image.Width <= width)
165                         {
166                             width = image.Width;
167                         }
168                     }
169
170                     int newHeight = image.Height * width / image.Width;
171                     if (newHeight > height)
172                     {
173                         // Redimensionar altura
174                         width = image.Width * height / image.Height;
175                         newHeight = height;
176                     }
177
178                     Image NewImage = image.GetThumbnailImage(width,
179                                                              newHeight,
180                                                              null,
181                                                              IntPtr.Zero);
182
183                     return NewImage;
184                 }
185             }
186             catch (Exception)
187             {
```

```
188                 return (Image)bmp; //Image.FromFile(file);
189             }
190         }
191
192         /// <summary>
193         /// trim the string and returns up to the specified length
194         /// </summary>
195         /// <param name="temp"></param>
196         /// <param name="length"></param>
197         /// <returns></returns>
198         public static string GetShortString(string temp, int length)
199         {
200             string newstring;
201             if (temp.Length > length)
202             {
203                 newstring = temp.Remove(length);
204                 newstring = newstring + "...";
205             }
206             else
207             {
208                 newstring = temp;
209             }
210             return newstring;
211         }
212                     //Obtem a data com o formato utilizado pelo PC
213         public static string GetShortDate(DateTime DateTime)
214         {
215             if (DateTime.ToString() != "1/1/0001 12:00:00 AM")
216             {
217                 return DateTime.ToString("ddd dd MMM yyyy",
                      CultureInfo.CurrentCulture);// hh:mm");//",
                      CultureInfo.InvariantCulture);
218             }
219             else
220             {
221                 return "";
222             }
223         }
224     }
225
226     public class ComboboxItem
227     {
228         public string Text { get; set; }
229         public object Value { get; set; }
230
231         public override string ToString()
232         {
233             return Text;
234         }
```

```
235     }
236 }
237
```

```
1 using System;
2 using System.Collections.Generic;
3
4 using System.Web;
5 using System.Data.SqlClient;
6 using System.Data;
7 using BibliotecaVirtual.BusinessLogic;
8
9 /// <summary>
10 /// Summary description -For DBHandler
11 /// </summary>
12 public class DBHandler
13 {
14                                   //Metodos para estabelecer communicação com a BD
                        e executar SP
15     private SqlConnection CreateConnection(string connString)
16     {
17         return new SqlConnection(connString); //cria ligação
18     }
19     private void CloseConnection(SqlConnection connection)
20     {
21         connection.Close();
22     }
23
24     public DataTable ExecuteSP(string SPName, List<SQLParameters> Parameters)
25     {
26         try     //Executa SP
27         {
28             DataSet dsData = ExecuteSPDatSet(SPName, Parameters);
29
30             i-F (dsData.Tables.Count > 0)
31                 return dsData.Tables[0];
32             else
33                 return null;
34         }
35         catch (Exception ex)
36         {
37
38             throw ex;
39         }
40     }
41
42
43
44     public DataSet ExecuteSPDatSet(string SPName, List<SQLParameters> Parameters)
45     {
46         try
47         {
48                     //Carrega parametros da connection string existentes no
```

```
                    -Ficheiro de con-Figurações
49              SqlConnection conn = CreateConnection(Helper.GetCon-FigValue
                  ("BibliotecaVirtual_Database"));
50
51              SqlCommand command = new SqlCommand(SPName, conn); //comando para
                  execução SP na DB
52              command.CommandType = CommandType.StoredProcedure;
53
54              i-F (Parameters != null)
55              {
56                  -Foreach (SQLParameters Parameter in Parameters)
57                  {
58
59                      i-F (Parameter.ParamValueBinary != null)
60                          command.Parameters.Add("@" + Parameter.ParamName,
                          Parameter.DBtype).Value = Parameter.ParamValueBinary;
61                      else
62                          command.Parameters.Add("@" + Parameter.ParamName,
                          Parameter.DBtype).Value = Parameter.ParamValue;
63                  }
64              }
65              DataSet dtData = new DataSet();
66
67              SqlDataAdapter adp = new SqlDataAdapter();
68              adp.SelectCommand = command;
69              adp.Fill(dtData);
70
71              CloseConnection(conn);
72
73              return dtData;
74          }
75          catch (Exception ex)
76          {
77              throw ex;
78          }
79      }
80  }
81  public class SQLParameters
82  {
83      public string ParamName;
84      public SqlDbType DBtype;
85      public string ParamValue;
86      public byte[] ParamValueBinary;
87  }
```

```
1  using BibliotecaVirtual_Users;
2  using System;
3  using System.Collections.Generic;
4  using System.ComponentModel;
5  using System.Data;
6  using System.Drawing;
7  using System.Linq;
8  using System.Text;
9  using System.Threading.Tasks;
10 using System.Windows.Forms;
11
12 namespace BibliotecaVirtual
13 {
14     public partial class frmUsers : Form
15     {
16         public frmUsers()
17         {
18             Inicializa_Componente();
19         }
20
21         private void Form1_Load(object sender, EventArgs e)
22         {
23             try
24             {              // Obtem o username do utilizador autenticado no
                  Dominio para ser apresentado na janela de pesquisa
25                 lblUserName.Text =
                     System.Security.Principal.WindowsIdentity.GetCurrent().Name;
26
27         List<SQLParameters> param = new List<SQLParameters>();
28               //adiciona ao parametro o username e a data/hora de
           login
29                param.Add(new SQLParameters { ParamName = "UserName", ParamValue =
           lblUserName.Text, DBtype = SqlDbType.VarChar });
30
31         DBHandler dbHandler = new DBHandler();                    //Instancia
           classe
32           DBHandler que contém os metodos de ligação á base de dados //Insere
           na base de dados o parametro username e a data/hora de
33         login, com a execução do package.
34         dbHandler.ExecuteSP("usr_UpdateVMUserLoginTime", param);
35         }
36         catch (Exception)
37         {
38
39         throw;
40         }
41         }
42
43         private void btnSearch_Click(object sender, EventArgs e)
           {
```

```
44              frmSearch search = new frmSearch(); //Instancia a classe frmSearch()
45              search.SearchText = txtSearch.Text; //Passa o termo da pesquisa
                  inserido na caixa de texto
46              search.Show();                      //para a classe que tem a função
                  de executar a pesquisa
47              Hide();
48          }
49
50          private void InitializeComponent()
51          {
52              this.SuspendLayout();
53              //
54              // frmUsers -
55              //
56              this.ClientSize = new System.Drawing.Size(284, 261);
57              this.Name = "frmUsers";
58              this.ResumeLayout(false);
59
60          }
61      }
62  }
63
```

```
 1 using BibliotecaVirtual.BusinessLogic;
 2 using BibliotecaVirtual_Users.Controls;
 3 using BibliotecaVirtual_Users.Properties;
 4 using System;
 5 using System.Collections.Generic;
 6 using System.ComponentModel;
 7 using System.Data;
 8 using System.Drawing;
 9 using System.Linq;
10 using System.Text;
11 using System.Threading.Tasks;
12 using System.Windows.Forms;
13
14 namespace BibliotecaVirtual_Users
15 {
16     public partial class frmSearch : Form
17     {
18         public string SearchText
19         {
20             set { txtSearch.Text = value; }
21             get { return txtSearch.Text; }
22         }
23
24
25         #region Form
26         public frmSearch()
27         {
28
29             Inicializa_Componente();
30         }
31
32         private void frmSearch_Load(object sender, EventArgs e)
33         {
34             try
35             {
36                 this.Text = "Procurar por : " + SearchText;
37                       //Instancia classe DBHandler que contém os metodos de
                          ligação á base de dados
38                 DBHandler dbHandler = new DBHandler();       //Instancia classe
                     DBHandler que contém os metodos de ligação á base de dados
39                       //Executa SP para
                          obter lista de categorias
40                 DataTable dtCats = dbHandler.ExecuteSP("bk_GetAllBookCategories",
                       null);
41
42                 foreach (DataRow row in dtCats.Rows)
43                 {
44                     ComboboxItem item = new ComboboxItem(); //Preenche combobox
                        com lista
```

```
45
46                  item.Text = row["CategoryName"].ToString();
47                  item.Value = row["CategoryID"].ToString();
48
49                  ddlCategories.Items.Add(item);
50              }
51                                  //Executa SP para obter lista de
                     todos os tipos de documentos.
52              DataTable dtTypes = dbHandler.ExecuteSP("bk_GetAllDocumentTypes",
                  null);
53
54              foreach (DataRow row in dtTypes.Rows)
55              {
56                  ComboboxItem item = new ComboboxItem(); //Preenche
                    combobox com lista
57
58                  item.Text = row["DocumentTypeName"].ToString();
59                  item.Value = row["DocumentTypeID"].ToString();
60
61                  ddlDocTypes.Items.Add(item);
62              }
63                              //Executa SP para obter lista de todas as
                    editoras
64              DataTable dtPublishers = dbHandler.ExecuteSP
                  ("bk_GetAllUniqueBookPublishers", null);
65
66              foreach (DataRow row in dtPublishers.Rows)
67              {
68                  ComboboxItem item = new ComboboxItem(); //Preenche
                    combobox com lista
69
70                  item.Text = row["PublisherInfo"].ToString();
71
72                  ddlPublishers.Items.Add(item);
73              }
74
75              ddlDocTypes.SelectedIndex = ddlCategories.SelectedIndex =
                  ddlPublishers.SelectedIndex = 0;
76                  //definição do tamanho do painel de pesquisa
77              tblSearchPanel.Size = new Size(this.Size.Width
                - 20, this.Size.Height);
78              tblSearchPanel.MaximumSize = new Size(this.Size.Width, 600);
79
80              SearchData(); //Executa a pesquisa -- Função mais abaixo.
81
82              this.WindowState = FormWindowState.Maximized; //Maximiza a
                  janela de pesquisa.
83
84          }
```

```
85              catch (Exception ex)
86              {
87                  MessageBox.Show(ex.Message);
88              }
89          }
90
91          private void frmSearch_Resize(object sender, EventArgs e)
92          {
93              try
94              {
95                  var loc = tblSearchBar.PointToScreen(Point.Empty);
96                  //Redimensiona o tamanho do painel de pesquisa
97                  tblSearchPanel.Top = loc.Y + tblSearchBar.Height + 30;
98
99                  tblSearchPanel.MaximumSize = new Size(this.Size.Width
                      - 30,
                      this.Size.Height - tblSearchBar.Height - 50);
100                 tblSearchPanel.Size = new Size(this.Size.Width - 30,
                      this.Size.Height - tblSearchBar.Height - 50);
101
102
103             }
104             catch (Exception)
105             {
106
107             }
108
109         }
110
111         private void frmSearch_FormClosed(object sender, FormClosedEventArgs
e)
112         {
113             Environment.Exit(0);
114         }
115         #endregion
116
117         #region Searching
118
119         private void SearchData()
120         {
121             try
122             {
123                 //Executa a pesquisa com base nos critérios selecionados.
124                 Application.DoEvents();
125                 tblSearchPanel.Controls.Clear();
126                 btnStartSearch.Text = "Procurando...";
127                 DBHandler dbHandler = new DBHandler(); //Instancia classe
                      DBHandler que contém os metodos de ligação á base de dados
128
```

```
129             List<SQLParameters> param = new List<SQLParameters>(); //Pesquisa
                    avançada, com base nas opções escolhidas (Titulo,
                  tipoDoc, ...)
130
131             param.Add(new SQLParameters { ParamName = "Title", ParamValue =
                  txtSearch.Text, DBtype = SqlDbType.VarChar });
132             param.Add(new SQLParameters { ParamName = "DocTypeID", ParamValue
                  = ddlDocTypes.SelectedIndex > 0 ? (ddlDocTypes.SelectedItem as
                  ComboboxItem).Value.ToString() : "", DBtype = SqlDbType.VarChar
                  });
133             param.Add(new SQLParameters { ParamName = "CategoryID",
                  ParamValue = ddlCategories.SelectedIndex > 0 ?
                  (ddlCategories.SelectedItem as ComboboxItem).Value.ToString() :
                  "", DBtype = SqlDbType.VarChar });
134             param.Add(new SQLParameters { ParamName = "PublisherInfo",
                  ParamValue = ddlPublishers.SelectedIndex > 0 ?
                  (ddlPublishers.SelectedItem as ComboboxItem).Text : "",
                  DBtype = SqlDbType.VarChar });
135                //executa SP e passa os parametros com os critérios
                    da pesquisa.
136             DataTable dtData = dbHandler.ExecuteSP("bk_SearchBooks", param);
137
138             if (dtData.Rows.Count < 0)
139             {
140                 MessageBox.Show("Não foram encontrados resultados. Utilize
                    termos diferentes.");
141             }
142             else
143             {
144                 foreach (DataRow row in dtData.Rows)
145                 {
146                    try
147                 { //Carrega o resultado da pesquisa -
                    Lista com Imagens das capas, e informação dos livros
148                 SearchedBook searchedBook = new SearchedBook
                    (Convert.ToInt64(row["BookID"].ToString()), row
                    ["Title"].ToString(), row["Authors"].ToString(), row
                    ["PublisherInfo"].ToString(), row["ReleaseDate"].ToString(),
149
      row["Abstract"].ToString(), !string.IsNullOrEmpty(row
    ["CoverImageData"].ToString()) ? Helper.ResizeImage((byte[])row
    ["CoverImageData"], 135, 180, true) : Helper.ResizeBitmapImage(Resources.NoImage,
    135, 180, true));
150
151                 searchedBook.Anchor = (AnchorStyles.Top |
                    AnchorStyles.Left | AnchorStyles.Right);
152
153                 tblSearchPanel.Controls.Add(searchedBook); //
                    apresenta pesquisa no painel da janela da aplicação.
154                    }
```

```
155                     catch (Exception)
```

```
156                         {
157                         }
158                     }
159                 }
160                 btnStartSearch.Text = "Procurar"; //Texto do botão
161             }
162             catch (Exception ex)
163             {
164                 btnStartSearch.Text = "Procurar";
165                 MessageBox.Show(ex.Message);
166             }
167         }
168 
169         private void btnStartSearch_Click(object sender, EventArgs e)
170         {
171             SearchData();  // Botão pesquisar
172         }
173 
174         #endregion
175 
176     }
177 }
178 
```

```
 1 using BibliotecaVirtual_Users.Properties;
 2 using System;
 3 using System.Collections.Generic;
 4 using System.ComponentModel;
 5 using System.Data;
 6 using System.Drawing;
 7 using System.Linq;
 8 using System.Text;
 9 using System.Threading.Tasks;
10 using System.Windows.Forms;
11
12 namespace BibliotecaVirtual_Users
13 {
14     public partial class frmBookDetails : Form
15     {
16         public frmBookDetails()
17         {
18             Inicializa_Componente();
19         }
20
21
22         public void LoadData(long BookID)
23         {
24             searchedBookDetails.LoadData(BookID);
25         }
26     }
27 }
28
```

```
1 using System;
2 using System.Collections.Generic;
3 using System.ComponentModel;
4 using System.Drawing;
5 using System.Data;
6 using System.Linq;
7 using System.Text;
8 using System.Threading.Tasks;
9 using System.Windows.Forms;
10 using BibliotecaVirtual.BusinessLogic;
11
12 namespace BibliotecaVirtual_Users.Controls
13 {
14     public partial class SearchedBook : UserControl
15     {
16         private long bookID;
17         public long BookID
18         {
19             set { bookID = value; }
20             get { return bookID; }
21         }
22
23         public Image CoverImage
24         {
25             set { pbCoverImage.Image = value; }
26         }
27         public string Title
28         {
29             set { lblTitle.Text = value; }
30         }
31         public string Author
32         {
33             set { lblAuthors.Text = value; }
34         }
35         public string Publisher
36         {
37             set { lblPublishedBy.Text = value; }
38         }
39         public string ReleaseDate
40         {
41             set { lblReleaseDate.Text = value; }
42         }
43         public string Abstract
44         {
45             set { lblAbstract.Text = value; }
46         }
47         public SearchedBook(long bookID, string title, string
           author, string
48         {           publisher, string releaseDate, string abstractText, Image
                                                          coverImage)
```

```
49              Inicializa_Componente();
50
51              BookID = bookID;
52              Title = title;
53              Author = author;
54              Publisher = publisher;
55              ReleaseDate = !string.IsNullOrEmpty(releaseDate) ? Helper.GetShortDate
          (DateTime.Parse(releaseDate))      :"";
56
57              Abstract = Helper.GetShortString(abstractText, 1000);
58
59              CoverImage = coverImage;
60
61
62          }
63
64          private void lblTitle_Click(object sender, EventArgs e)
65          {
66              frmBookDetails details = new frmBookDetails();
67              details.LoadData(BookID);
68              details.ShowDialog();
69          }
70
71          private void SearchedBook_Resize(object sender, EventArgs e)
72          {
73              lblAbstract.MaximumSize = new Size(tableLayoutPanel3.Width, 0);
74              lblAbstract.AutoSize = true;
75
76          }
77
78          private void btnDetails_Click(object sender, EventArgs e)
79          {
80              frmBookDetails details = new frmBookDetails();
81              details.LoadData(BookID);
82              details.ShowDialog();
83          }
84
85
86      }
87  }
88
```

```
 1 using System;
 2 using System.Collections.Generic;
 3 using System.ComponentModel;
 4 using System.Drawing;
 5 using System.Data;
 6 using System.Linq;
 7 using System.Text;
 8 using System.Threading.Tasks;
 9 using System.Windows.Forms;
10 using BibliotecaVirtual.BusinessLogic;
11 using BibliotecaVirtual_Users.Properties;
12 using System.IO;
13
14 namespace BibliotecaVirtual_Users.Controls
15 {
16     public partial class SearchedBookDetails : UserControl
17     {
18         private byte[] pdfData;
19
20         public Image CoverImage
21         {
22             set { pbCoverImage.Image = value; }
23         }
24         public string Title
25         {
26             set { lblTitle.Text = value; }
27             get { return lblTitle.Text; }
28         }
29         public string Author
30         {
31             set { lblAuthors.Text = value; }
32         }
33         public string Publisher
34         {
35             set { lblPublishedBy.Text = value; }
36         }
37         public string ReleaseDate
38         {
39             set { lblReleaseDate.Text = value; }
40         }
41         public string Abstract
42         {
43             set { lblAbstract.Text = value; }
44         }
45         public SearchedBookDetails()
46         {
47             Inicializa_Componente();
48
49             tableLayoutPanel3.RowCount = 0;
```

```
50              tableLayoutPanel3.RowStyles.Clear();
51              tableLayoutPanel3.AutoScroll = true;
52          }
53
54          public void LoadData(long BookID)
55          {
56              try
57              {
58                  DBHandler dbHandler = new DBHandler();    //Instancia classe
                      DBHandler que contém os metodos de ligação á base de dados
59
60                  List<SQLParameters> param = new List<SQLParameters>();
61                        //Adiciona parametro com ID do livro
62                  param.Add(new SQLParameters { ParamName = "BookID", ParamValue =
                      BookID.ToString(), DBtype = SqlDbType.BigInt });
63                        //Executa SP com o ID do documento como
64                      parametro DataTable dtBookDetails =
                      dbHandler.ExecuteSP ("bk_GetBookDetails", param);
65
66                  i-F (dtBookDetails.Rows.Count > 0)
67                  {
68                      DataRow row = dtBookDetails.Rows[0];
69                        //Obtem detalhes sobre a in-Formação do livro
70                      Title = row["Title"].ToString();
71                      Abstract = row["Abstract"].ToString(); CoverImage =
72                      !string.IsNullOrEmpty(row ["CoverImageData"].ToString())
                        ? Helper.ResizeImage((byte[]) row["CoverImageData"],
                        135, 180, true) :
                        Helper.ResizeBitmapImage(Resources.NoImage, 135, 180,
73                      true); ReleaseDate = !string.IsNullOrEmpty(row
                        ["ReleaseDate"].ToString()) ? Helper.GetShortDate
                        (DateTime.Parse(row["ReleaseDate"].ToString())) : "";
74                      Publisher = row["PublisherIn-Fo"].ToString(); Author =
75                      row["Authors"].ToString();
76
77                      pd-FData = (byte[])row["PDFData"];
78                  }
79              }
80              catch (Exception ex)
81              {
82                  MessageBox.Show(ex.Message);
83              }
84          }
85
86          private void picDownloadFile_Click(object sender, EventArgs e)
87          {
88              try
89              {
90                  i-F (pd-FData != null)
```

```
91              {
92                   svFile.FileName = Title + ".pd-F";
93                   i-F (svFile.ShowDialog() ==
                     System.Windows.Forms.DialogResult.OK)
94                  {
95                 File.WriteAllBytes(svFile.FileName, pd-FData);
96
97                MessageBox.Show("O -Ficheiro PDF -Foi guardado com sucesso
                      em:" + svFile.FileName);
98                  }
99              }
100             else
101                 MessageBox.Show("Ficheiro PDF não encontrado.");
102         }
103         catch (Exception)
104         {
105
106             throw;
107         }
108     }
109 }
110
111
```