Estado da Arte Projeto Global

Biblioteca Virtual

Aluno: Célio Faustino
Nº: 1887

Coordenador: Prof. Dr. Pedro Brandão

Lisboa 2015/2016

Índice

Índice de Imagens

Resumo

Neste estado da arte vou abordar as tecnologias aplicadas neste projeto. Irá ser desenvolvido um projeto onde irá ser criada uma Biblioteca Virtual com acesso a uma base de dados, usando o potencial das novas tecnologias, tendo em conta as vantagens e desvantagens.

Com o aparecimento da Internet, foram criadas novas tecnologias, tal como a *Cloud Computing* (ou computação em nuvem), a virtualização, SQL[1] e ADO.NET[2]. Essas tecnologias permitiram a vários setores sociais e empresariais trabalharem cooperativamente, tanto em instituições públicas como educacionais. São essas tecnologias que vou abordar e usar neste projeto.

Cloud Computing é um modelo para permitir, a pedido, acesso à rede de forma ubíqua, conveniente a uma *pool* compartilhada de recursos de computação configuráveis (por exemplo, redes, servidores, armazenamento, aplicações e serviços) que podem ser rapidamente fornecidos e liberados com esforço de gerenciamento mínimo ou interação do provedor de serviço.

Virtualização é uma abstração representada por um recurso computacional que oferece um ambiente completo, similar ao de uma máquina física, com sistema operativo, aplicações e serviços de rede. É mais conhecida por máquina virtual.

O *SQL* (*Structured Query Language*) é uma linguagem de gestão e manipulação de dados relacionais através de programas de SGBD (Sistema de Gestão de Base de Dados). Esses sistemas permitem trabalhar vários tipos de bases de dados: SQL Server, Access, Oracle, MySQL, etc.

O ADO.NET (*ActiveX Data Objects .NET*) é uma tecnologia baseada na plataforma .NET através de um conjunto de classes. A estrutura dos seus componentes foi desenhada para facilitar a manipulação e o tratamento de vários tipos de dados relacionais, XML e dados de aplicações.

Palavras-chave: Computação em nuvem; virtualização; base de dados.

[1] Linguagem de consulta estruturada (*Structured Query Language*)
[2] Serviço de acesso a dados para programadores

Abtract

In this state of the art I will address the technologies applied in this project. A virtual library with access to a database will be developed in a project, which will be created by using the potential of the new technologies taking into account the pros and cons.

With the advent of the Internet, new technologies were created, such as cloud computing, virtualization, SQL and ADO.NET. These technologies have enabled the various social and business sectors to work cooperatively, both in public and educational institutions. These are the technologies that I will address and use in this project.

Cloud Computing is a model to allow, on demand, ubiquitous and convenient access to the network, to a shared pool of configurable computing resources (e.g., networks, servers, storage, applications and services) that can be rapidly provisioned and released with minimal management effort or service provider interaction.

Virtualization is an abstraction represented by a computational resource that provides a complete environment, similar to a physical machine, with operating system, applications and network services. It is best known as a virtual machine.

SQL (Structured Query Language) is a management and manipulation language of relational data through DBMS programs (Database Management System). These systems allow you to work with various types of databases: SQL Server, Access, Oracle, MySQL, etc.

The ADO.NET (ActiveX Data Objects .NET) is a technology based on the .NET platform through a set of classes. The structure of the components was designed to facilitate the handling and treatment of various types of relational data, XML and applications data.

Keywords: Cloud Computing; virtualization; data base.

Introdução

"Uma biblioteca digital ou virtual é uma biblioteca em que uma proporção significativa de recursos de informação se encontram disponíveis em formato digital (pdf, doc, etc), acessível por meio de computadores. É importante considerar que o conceito de biblioteca virtual está presente no efeito de integração da informática com as comunicações, cujo expoente máximo é a internet." (Alemán, 2010).

O presente estado da arte apresenta, para o projeto em curso, uma introdução das tecnologias utilizadas e que tiram partido do potencial das comunicações em rede. As tecnologias para o desenvolvimento deste projeto serão: *Cloud Computing*, virtualização, SQL e ADO.NET. Será criada uma aplicação de uma biblioteca virtual que permitirá interagir com máquinas virtuais onde estará a base de dados.

Neste trabalho abordarei os vários conceitos e a sua importância, expondo as vantagens e as desvantagens.

ESTADO DA ARTE

Cloud Computing

1. Referencial teórico

Executar aplicações, guardar e aceder a dados armazenados na *cloud*, via internet, numa componente de oferta de serviços, ofereceu várias vantagens após o aparecimento do conceito de *Cloud Computing* (Armbrust, 2009).

Com o aparecimento do termo *Cloud Computing* numa palestra de Eric Schmidt, diretor executivo da empresa Google, o mesmo falava sobre como a empresa descrevia a sua abordagem do *Software*[3] como um serviço. Atualmente, *Cloud Computing*, ou Computação na Nuvem, traz profundas mudanças no mundo das tecnologias de informação (Jozwiak & Bloor, 2011).

Segundo Dan C. Marinescu, "A computação em nuvem usa tecnologias da internet para oferecer serviços elásticos. O termo "computação elástica" refere-se à capacidade de adquirir dinamicamente os recursos de computação e suportar uma carga de trabalho variável. Um provedor de serviço de nuvem mantém uma infraestrutura maciça para apoiar os serviços elásticos" (Marinescu, 2013).

No próximo ponto será abordado o conceito *Cloud Computing*. A sua importância e a sua aplicação, expondo as vantagens e as desvantagens no uso deste tipo de serviços.

2. Conceito de *Cloud*

"*Clouds* são plataformas tecnológicas que aproveitam as inovações sofisticadas da tecnologia e proporcionam ambientes altamente escaláveis e flexíveis que podem ser remotamente utilizados por organizações numa poderosa infinidade de maneiras. Para

[3] Sequência de instruções escritas para serem interpretadas por um computador com o objetivo de executar tarefas específicas.

construir com sucesso, integrar-se, ou até mesmo criar um ambiente de nuvem requer uma compreensão da sua mecânica interior, camadas de arquitetura e modelos, bem como uma compreensão do negócio e fatores económicos que resultam da adoção e da utilização de serviços baseados em nuvem no mundo real." (Erl, Puttini, & Mahmood, 2013).

Cloud Computing não é mais do que ter à disposição uma capacidade de computação flexível e que está disponível através de uma ligação à rede (Cooter, 2011).

Os serviços de *Cloud* são disponibilizados através de uma entidade que se responsabiliza pelo processo de processamento e armazenamento dos dados, largura de banda, fiabilidade e segurança (em *datacenters* criados para o efeito), e presta serviços, alocando e liberando recursos, conforme as necessidades do utilizador. O utilizador pode utilizar os serviços pagando apenas pelo que usa e gastar o menos possível, mas se no seu negócio previr um pico de vendas, pode aumentar a capacidade de processamento e largura de banda de modo a não perder clientes e voltar ao serviço que tinha após o pico. O utilizador irá pagar mais nesse período de pico porque solicitou mais capacidade ou processamento ou largura de banda (Bent & Steeg, 2012).

As organizações conseguem reconhecer que a *Cloud Computing* é um meio lucrativo e que têm a tecnologia mais recente do mundo da computação. A *cloud* beneficia as organizações a nível operacional, económico, na segurança e na facilidade na colaboração (Rao, 2015).

Como qualquer outra tecnologia, o *Cloud Computing* tem as suas vantagens e desvantagens no que toca às aplicações, negócios, pesquisas, desenvolvimentos ou outra finalidade.

3. Vantagens da utilização do *Cloud Computing*

O benefício mais evidente será o da redução dos custos operacionais e de manutenção com a utilização da computação em sistemas de nuvem. Esse benefício é representado pelo ROI (*Return Of Investment*) que é a relação entre os lucros e os prejuízos num investimento. As empresas ou organizações que utilizam os serviços de *cloud* apenas pagam pelo software e infraestrutura que estão a utilizar. Esta é uma grande vantagem para novas empresas ou organizações que estejam a entrar no mundo dos negócios e não têm capital suficiente para investir em equipamento informático e serviços

administrativos. Para um operacional de uma empresa, o ter os seus dados e serviços sempre disponíveis em qualquer lugar e momento, via internet, através de vários dispositivos como portáteis, *smartphones* ou *tablets* também é uma vantagem a ter em conta. Concentrando todos esses serviços e infraestruturas de TI[4] em *datacenters*, também otimiza consideravelmente a afetação de recursos e eficiência energética. Isso pode reduzir bastante o impacto sobre o meio ambiente (Buyya, Vecchiola, & Selvi, 2013).

4. Desvantagens da utilização do *Cloud Computing*

Segundo Barry Sosinsky, existem muitas vantagens nos serviços disponibilizados na nuvem, mas é necessário ter em conta que a privacidade e a segurança podem ser uma desvantagem, apesar do esforço para garantir padrões de fiabilidade e que sejam reconhecidos internacionalmente. Ainda não há muita clareza e consenso nos direitos e deveres no que diz respeito às partes envolvidas no uso dos modelos de *Cloud Computing*. Barry Sosinsky diz ainda que poderão existir necessidades de maior latência de rede ao tentar transferir uma quantidade elevada de dados ou usar aplicações que, para a sua execução necessitem de *hardware* robusto (Sosinsky, 2011).

Os serviços *cloud* também estão sujeitos às leis e são obrigados a atuar conforme a legislação vigente em cada país onde se localizam os *datacenters*. Isso pode ter impacto nos serviços de muitas empresas (Catteddu & Hogben, 2009).

[4] Tecnologias de Informação

1. Referencial teórico

Podemos definir a virtualização atualmente como uma metodologia que permite a divisão dos recursos de um computador entre múltiplos ambientes de execução. Peter Baer Galvin refere que, num sistema virtualizado, temos um *software host*[5] em execução numa máquina física chamado *Hypervisor*[6] ou Virtual Machine Monitor (VMM) e é responsável pela criação de VM's (*Virtual Machines*) que são ambientes simulados de computação. Assim pode se executar Sistemas Operativos em máquinas virtuais como se estivessem instalados numa máquina real (Galvin, 2009).

Segundo João Almeida da IBM, num artigo da Computer World, em outubro de 2011, ocorreu uma renovação de *hardware* antigo com o objetivo de renovar a infraestrutura de TI (Almeida, 2011). Isto deve-se à evolução dos sistemas de hardware, ao aumento da capacidade de processamento, memória e disco por forma a executar mais tarefas em simultâneo e com um custo cada vez menor, fazendo com que a virtualização aparecesse em maior escala nos últimos anos.

Pode dizer-se que a virtualização nasceu em 1959 com a publicação do artigo "Time Sharing Processing in Large Fast Computers" numa conferência internacional de processamento de informação que se realizou em Nova York. O artigo estava concentrado no uso da multiprogramação[7] e no conceito das máquinas de grande porte onde se poderia utilizar melhor os recursos de *hardware* (Strachey, 1959).

A IBM introduziu posteriormente o conceito de multiprocessamento[8] e memória virtual[9] como parte do sistema operativo nos *mainframes*. Isso permitiu que vários processadores trabalhassem como um só, possibilitando a abstração e o mapeamento da memoria real para memória virtual e a especificação de partições que eram utilizadas por programas diferentes, surgindo assim as primeiras formas de virtualização.

---

[5] Hospedeiro
[6] Um monitor de máquinas virtuais que cria e executa máquinas virtuais
[7] Técnica de desenvolvimento e execução de dois ou mais programas em simultâneo numa máquina
[8] Capacidade de um sistema operacional executar simultaneamente dois ou mais processos
[9] Técnica que usa a memória secundária como uma cache para armazenamento secundário

2. Vantagens da Virtualização

Na virtualização, o isolamento que se consegue ter entre as máquinas e o sistema operativo anfitrião faz com que algumas tarefas arriscadas sejam seguras. Normalmente a avaliação de novos sistemas, desenvolvimentos de *software*, testes com vírus e outras ameaças significa sujeitar o computador a operações, onde nem sempre é possível reverter os problemas (Goldberg, 1974). É também bom para a consolidação de servidores, reduzir o consumo de energia elétrica em *datacenters*, menor produção de calor, espaço físico e manutenção de equipamentos, com menos custos e menor degradação (Marshall, 2011).

A utilização da virtualização, tem-se relevado uma alternativa interessante em diversos modelos da computação ao longo dos anos. Entre eles estão a consolidação e centralização de servidores, otimizações de hardware e a segurança da informação. Os administradores de TI das empresas beneficiam muito com as estruturas virtuais porque destinam os recursos dos servidores estrategicamente conforme a necessidade do negócio (Okano & Andrade, 2008).

Com a consolidação de servidores, as empresas reduzem o número de servidores que necessitam de manutenção evitando custos administrativos e operacionais. Em vez de utilizarem vários servidores com os respetivos sistemas operativos, utilizam um servidor com máquinas virtuais e os respetivos sistemas operativos independentes com aplicações e serviços instalados (Mann, 2006).

3. Desvantagens da Virtualização

Apesar de toda a publicidade sobre a virtualização, pode haver desvantagens ao migrar para um ambiente virtualizado. As aplicações que são executadas num ambiente físico usam mais recursos e não têm um melhor desempenho num ambiente virtualizado. A não ser que a migração também inclua a renovação de novo hardware que seja mais rápido e potente, as aplicações podem ter menor desempenho (Gilster, 2014).

Segundo Manoel Veras, "A virtualização inicialmente utilizou o conceito de máquina virtual de processo. Uma máquina virtual de processo é uma aplicação que é executada num sistema operacional A e que emula o comportamento do sistema operacional B. Assim, aplicações desenvolvidas para o sistema operacional B podem ser executadas num sistema operacional A. É importante salientar que essa técnica de

implementação permite que binários de um processador sejam interpretados e substituídos por código equivalente de outro processador. Portanto, além de emular, o sistema operacional, é possível emular processadores. As desvantagens dessa técnica são basicamente duas: desempenho e desperdício de capacidade do *hardware* físico. O desempenho é sacrificado, já que há uma tradução de um sistema para o outro. O desperdício da capacidade física do *hardware* vem do fato que as máquinas virtuais de processo oferecem dispositivos de I/O genéricos e simples e este fato sobrecarrega o sistema operacional". Manoel Veras diz ainda que "os monitores de máquinas virtuais VMM (*Virtual Machine Monitors*) ou *Hypervisors* surgiram para resolver as desvantagens das máquinas virtuais de processos. Eles são implementados como uma camada de *software* entre o *hardware* e o sistema operacional, oferecendo uma máquina virtual para o sistema operacional. São mais eficientes por tratar de outra forma os dispositivos I/O" (Veras, 2011).

ADO.NET

1. Referencial teórico

Para Jayden Ky, o ADO.NET é uma plataforma de acesso e manipulação de dados da Microsoft que suporta vários formatos de dados, incluindo dados guardados em bases de dados relacionais ou como XML. Esta tecnologia tem semelhanças com a antiga tecnologia ADO (*Access Data Objects*) mas pouco tem a ver com o seu antecessor. Existe desde a versão 1.0 da plataforma .NET e está disponível para qualquer linguagem de programação, incluindo C#. O ADO.NET disponibiliza uma maneira uniforme de aceder a bases de dados relacionais diferentes. Cada uma usa o seu próprio protocolo e, sem o uso do ADO.NET, poderá requerer escrever código diferente para cada tipo de base de dados. O ADO.NET disponibiliza um conjunto de classes para diferentes tipos de protocolos para acesso às bases de dados. Estas são chamadas de *Data Providers*[10] (Ky, 2013).

O ADO.NET é uma coleção de classes que atuam como um intermediário de bases de dados, tal como Access ou XML, e os consumidores de dados, tal como uma página de ASP.NET (*Active Server Page.NET*) (Millington & Kauffman, 2009).

2. Características

A arquitetura do ADO.NET está separada em duas esferas fundamentais: conectado e desconectado. A parte conectada representa os objetos que mantêm uma ligação para interagir com a base de dados. A única maior exceção é o objeto *DataAdapter* que atua como uma sentinela entre os modos conectado e desconectado (Malik, 2007).

Os objetos principais do modo conectado são: *Connection*, *Transaction*, *DataAdapter*, *Command*, *Parameter* e *DataReader*.

Tipo de Objeto	Descrição
Connection	Possibilita a conexão com a base de dados.

[10] Provedores de dados

	Exemplos de objetos: *OleDbConnection*, *SqlConnection*, *OracleConnection* e outros.
Transaction	Possibilita a execução de um grupo de comandos como um grupo ou como uma operação atômica. Se toda a operação correr bem as alterações serão feitas na base de dados, se não correr bem, toda a operação é desfeita. Exemplos de objetos: *OleDbTransaction*, *SqlTransaction*, *OracleTransaction* e outros.
DataAdapter	Este objeto atua como um intermediário entre os objetos conectados e desconectados do ADO.NET. Estabelece a conexão ou, dada uma conexão estabelecida, tem informações suficientes para lhe permitir compreender os dados de um objeto desconectado e agir de acordo com o banco de dados de uma forma pré-especificada. Exemplos de objetos: *OleDbDataAdapter*, *SqlDataAdapter*, *OracleDataAdapter* e outros.
Command	Este objeto representa um comando executável na fonte de dados subjacente. Este comando pode ou não retornar resultados e podem ser usados para manipular, consultar, atualizar ou até mesmo apagar os dados existentes. Este objeto também pode ser usado para manipular estruturas de tabelas subjacentes. Exemplos de objetos: *OleDbCommand*, *SqlCommand*, *OracleCommand* e outros.
Parameter	Um comando precisa de ser capaz de aceitar parâmetros. Isto permite que os comandos sejam mais flexíveis e aceitem valores de entrada e atuem em conformidade. Estes parâmetros podem ser de entrada, de saída ou retornar valores de procedimentos armazenados. Exemplos de objetos: *OleDbParameter*, *SqlParameter*, *OracleParameter* e outros.
DataReader	Este objeto é o equivalente a um cursor somente de leitura ou somente de encaminhamento firehose que lhe permite buscar dados de um banco de dados a uma velocidade extremamente alta, mas em um modo de forward-only e somente leitura. Exemplos de objetos: *OleDbDataReader*, *SqlDataReader*, *OracleDataReader* e outros.

Os objetos principais do modo desconectado são: *DataSet*, *DataTable*, *DataColumn*, *DataRow*, *DataView*, *Constraint* e *DataRelation*.

Tipo de Objeto	Descrição
DataSet	Este objeto é o núcleo central do modo desconectado do ADO.NET. Um DataSet também pode ser visto como uma coleção lógica de *DataTables* e *DataRelations*. Exemplos de objetos: *OleDbDataSet*, *SqlDataSet*, *OracleDataSet* e outros.
DataTable	Este objeto é muito similar a uma tabela de uma base de dados. É composto por colunas (*DataColumns*), registos (*DataRows*) e regras (Constraints). Exemplos de objetos: *OleDbDataTable*, *SqlDataTable*, *OracleDataTable* e outros.
DataColumn	Este objeto representa a coluna de uma tabela. Exemplos de objetos: *OleDbDataColumn*, *SqlDataColumn*, *OracleDataColumn* e outros.
DataRow	Este objeto representa uma linha de registos de uma tabela. Exemplos de objetos: *OleDbDataRow*, *SqlDataRow*, *OracleDataRow* e outros.
DataView	Este objeto permite criar uma visualização de uma *DataTable* baseado num filtro. Exemplos de objetos: *OleDbDataView*, *SqlDataView*, *OracleDataView* e outros.
Constraint	Este objeto permite criar restrições de chaves estrangeiras (*ForeignKeyConstraint*) e de valores únicos (*UniqueConstraint*) baseados nos dados da tabela. Exemplos de objetos: *OleDbConstraint*, *SqlConstraint*, *OracleConstraint* e outros.
DataRelation	Este objeto permite especificar relações entre várias tabelas e validar dados entre tabelas.

Exemplos de objetos: *OleDbDataRelation*, *SqlDataRelation*, *OracleDataRelation* e outros.

3. Vantagens do ADO.NET

Como está referenciado no livro "*A Programmer's Guide to ADO.NET in C#*", o ADO.NET é uma plataforma com classes orientada a objetos que tem diferentes provedores de dados com o mesmo modelo de programação. Cada um desses modelos trabalha com protocolos de bases de dados diferentes mas funcionam da mesma forma, apenas terá de alterar no código os *namespaces*[11] (espaços de nomes) e as configurações de conexão às mesmas. Se trabalha bem com um provedor de base de dados, por exemplo *SqlClient*, trabalha também bem com outros (Chand & Gold, 2002).

O ADO.NET tira partido de todos os serviços fornecidos pela *Framework*[12], tais como gestão automática de memória, *Garbage Collection*[13] entre outros, o que torna o ADO.NET fácil de manipular e o programador não tem que se preocupar com a gestão da memória usada pelo ADO.NET. Segundo Nuno Ferreira, as vantagens do ADO.NET passam pela existência de um recurso importante para a criação de aplicações web e com arquitetura multicamada, que é o suporte para base de dados desconectados. A grande vantagem é o poder se ligar à base de dados apenas quando é necessário, os recursos podem ser usados por outros utilizadores e aumenta a escalabilidade e desempenho das aplicações (Ferreira, 2004).

4. Desvantagens do ADO.NET

Quando uma aplicação acede à base de dados em modo desconectado, os recursos não são mantidos no servidor. O utilizador, ao alterar os dados, pode gerar conflitos uma vez que esses dados podem ter sido já modificados por outros utilizadores. Outro problema é a necessidade de haver constantes ligações à base de dados (Ferreira, 2004).

---

[11] São usados para organizar as diversas classes na programação na plataforma .NET

[12] Abstração que une códigos comuns entre vários projetos de software provendo uma funcionalidade genérica

[13] Gere a alocação e liberação de memória para a aplicação

SQL

1. Referencial teórico

As bases de dados surgiram nos anos 60 graças às investigações da IBM e outras companhias. Essas investigações foram mais centradas na automação para escritórios, automatização de armazenamento de dados e indexar tarefas que requeriam demasiado trabalho manual. Um pioneiro no campo da tecnologia em bases de dados foi Charles W. Bachman, que recebeu um prémio Turing em 1973. Embora a IBM tenha sido um líder em investigações de bases de dados, a Honeywell Information Systems, Inc., disponibilizou uma versão comercial do produto no ano de 1976, baseado nos mesmos princípios, mas foi desenhado e implementado separadamente da IBM. No princípio dos anos 80, companhias como a IBM e a Oracle, construíram as primeiras bases de dados baseadas no padrão SQL. A Oracle construiu a Oracle Version 2 e a IBM, mais tarde, construiu a SQL/DS. Outras empresas viriam também a construir os seus próprios sistemas com base no mesmo padrão (Wilton & Colby, 2005).

2. Vantagens do SQL

Segundo Paul Wilton e John Colby no livro "Beginning SQL", a principal vantagem é a rápida e eficiente obtenção dos dados. Os programas de gestão e manutenção das bases de dados, os chamados DBMS (Database Management System), são aperfeiçoados de forma a permitir a obtenção dos dados que queremos, e da maneira que queremos. As bases de dados ajudam a organizar os dados de uma maneira lógica e permite separar os dados em partes específicas. As bases de dados relacionais têm a vantagem adicional de permitir que especifique o quão diferente os dados estão relacionados entre si. As regras de consistência das bases de dados também são importantes para quando se adiciona, modifica ou elimina dados. Assim pode-se minimizar redundâncias (Wilton & Colby, 2005).

O SQL é uma linguagem de alto nível que oferece um maior grau de abstração do que as linguagens processuais. O programador pode assim especificar quais os dados

necessários, mas não precisa especificar como os obter. As aplicações escritas com SQL podem ser facilmente executadas noutros sistemas. O SQL, como uma linguagem, é independente na maneira como é implementada internamente e, embora seja simples e fácil de aprender, consegue lidar com situações complexas. Uma consulta à base de dados retorna o mesmo resultado, não olhando para como a otimização foi feita, com ou sem índices. O SQL não é meramente uma linguagem de consultas, é também uma linguagem que pode definir a estrutura e controlo de acesso aos dados, assim como inserir, modificar ou eliminar dados (Leon & Leon, 1999).

3. Desvantagens do SQL

Christopher J. Date, no seu livro "*Relational Database: Selected Writings*", referiu que a linguagem SQL está longe de ser uma linguagem relacional pelo fato de não existir ortogonalidade nas expressões, funções embutidas, variáveis indicadoras, constante *NULL*[14], referência a dados correntes, conjuntos vazios, falta de algumas funções, definição formal da linguagem após a sua criação, não dá suporte a alguns aspetos do modelo relacional (*join*[15] explícito, domínios, etc.), discordância com as linguagens hospedeiras, que são geralmente processuais e orientadas para registos e não para conjuntos (Date, 1986).

Outro aspeto negativo envolve consultas em que os dados não tenham uma descrição textual adequada. Consultas com referências a imagens só podem ser criadas através de um texto que as identifique, através de uma descrição ou do nome de arquivo.

[14] Representa um valor nulo
[15] Junção de duas ou mais tabelas

Digital Library

A aplicação Digital Library foi desenvolvida com a intenção de proporcionar um ambiente de biblioteca virtualizado, onde são guardados livros, imagens, artigos de jornal, entre outros num formato digital.

Os utilizadores desta aplicação são configurados como administradores ou como utilizadores da biblioteca virtual. Os utilizadores, ao iniciar a aplicação, são logo apresentados com dois tipos de ambiente: acesso total à aplicação por parte dos administradores e acesso restrito, com apenas opções de consulta e transferência de ficheiros, por parte dos utilizadores.

Esta aplicação foi desenvolvida com o *software* Microsoft Visual Studio Community 2015, na linguagem C# e o acesso à base de dados com a tecnologia ADO.NET. Irá ser utilizada num ambiente de máquinas virtuais, servidores e clientes, que serão explicadas detalhadamente no próximo capítulo.

As máquinas virtuais que fazem parte desta solução são as seguintes:

- Um servidor Windows Server Standard Edition com o nome DC1. Este servidor é o controlado do domínio e também é servidor de DHCP.
- Um servidor Windows Server Datacenter Edition com o nome SQL. Este servidor tem o software Microsoft SQL Server 2012 Enterprise Edition instalado, onde irá estar alojada a base de dados da aplicação.
- Um servidor Windows Server Datacenter Edition com o nome VMM. Este servidor tem o Hyper-V instalado para … onde está instalado o System Center Virtual Machine Manager para a gestão de servidores virtuais.
- Uma máquina cliente com o Windows 8.1 Professional com o nome VMM com a aplicação Digital Library instalada, onde os utilizadores fazem a consulta e transferência dos ficheiros.

Criação das Máquinas Virtuais

1. Arranque por Disco Virtual VHD

O ambiente de virtualização foi criado num Windows Server 2012 R2 com arranque por um disco virtual do tipo VHD.

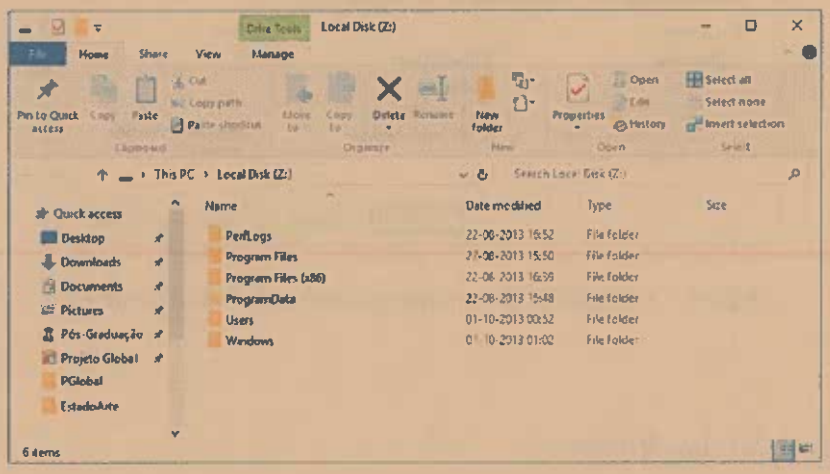

Figura 1 - Disco VHD com Windows Server 2012 R2 exposto como disco Z

O menu de arranque foi configurado, utilizando o caminho Z:\Windows

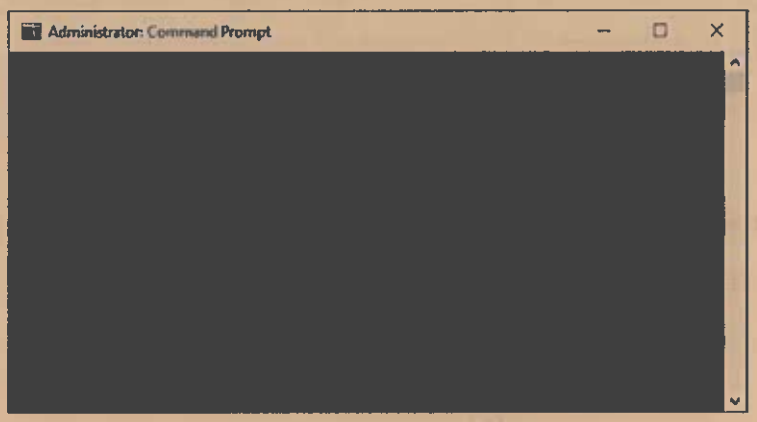

Figura 2 - Comando de boot

De seguida foi colocado o sistema operativo nativo de fábrica como o *Default*. Assim podemos escolher iniciar o computador tanto com o Windows pré-instalado,

neste caso o Windows 10, como com o Windows Server 2012 R2 que irá servir para a criação do ambiente.

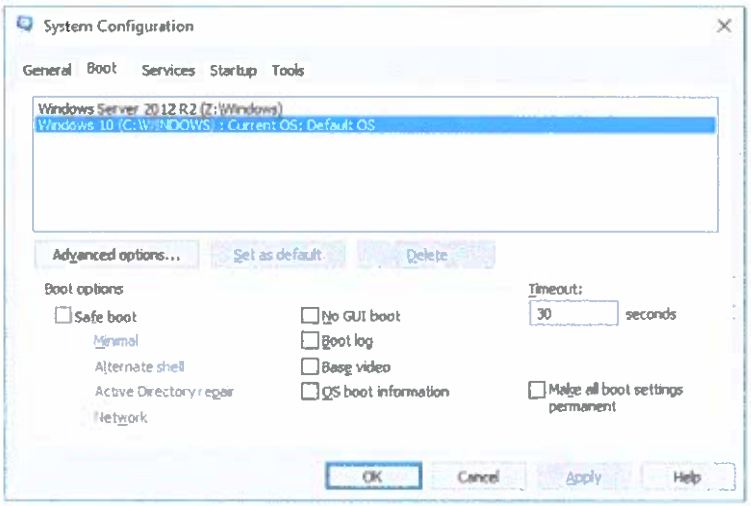

Figura 3 - Configuração de arranque de ambos os Sistemas Operativos

1. Configuração das Redes

A comunicação entre as máquinas foi configurada da seguinte forma:

VM	Função	Domínio	Tipo Rede VMWare	Rede	Segmento de Rede	IP	Subnet Mask	Gateway	DNS	Observações
DC1 Standard Edition	Domain Controller Obs: 1 vCPU, 1GB RAM	DigitalLib.local	NAT	NAT	(auto)	(auto)	(auto)	(auto)	(auto)	DHCP
			Host-Only	MGMTNET	10.1.1.0	10.1.1.100	/16 (255.255.0.0)	10.1.1.2	127.0.0.0	
SQL DataCenter Edition	SQL Server Enterprise Edition Obs: 1 vCPU, 2GB RAM	DigitalLib.local	NAT	NAT	(auto)	(auto)	(auto)	(auto)	(auto)	DHCP
			Host-Only	MGMTNET	10.1.1.0	10.1.1.101	/16 (255.255.0.0)	10.1.1.2	10.1.1.100	
			Host-Only	SQLNET	10.2.1.0	10.2.1.101	/16 (255.255.0.0)	(vazio)	(vazio)	
VMM DataCenter Edition	Virtual Machine Manager Obs: 1 vCPU, 4GB RAM	DigitalLib.local	NAT	NAT	(auto)	(auto)	(auto)	(auto)	(auto)	DHCP
			Host-Only	MGMTNET	10.1.1.0	10.1.1.102	/16 (255.255.0.0)	10.1.1.2	10.1.1.100	
			Host-Only	SQLNET	10.2.1.0	10.2.1.102	/16 (255.255.0.0)	(vazio)	(vazio)	

Figura 4 - Configuração das Redes

Foram criados utilizadores de serviço para serem utilizados nas aplicações SQL Server e no System Center Virtual Machine Manager.

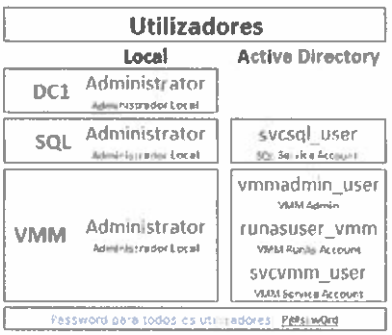

Figura 5 - Utilizadores

2. Criação das máquinas virtuais

a) DC1 - Controlador de Domínio

Para a criação da máquina virtual que irá ser o controlador do domínio para este ambiente, clicamos no botão "*Create a new Virtual Machine*" na aplicação VMware Workstation 12.

Figura 6 - Consola VMware Workstation 12

A criação de máquinas neste ambiente (DC1 para o controlador de domínio, SQL para o alojamento da base de dados da aplicação e VMM para a criação de máquinas virtuais cliente), irá ser da seguinte forma, com a diferença da chave do produto, da versão a instalar e da memória atribuída a cada máquina virtual.

Figura 7 - Criação em modo avançado para permitir configurar a máquina virtual

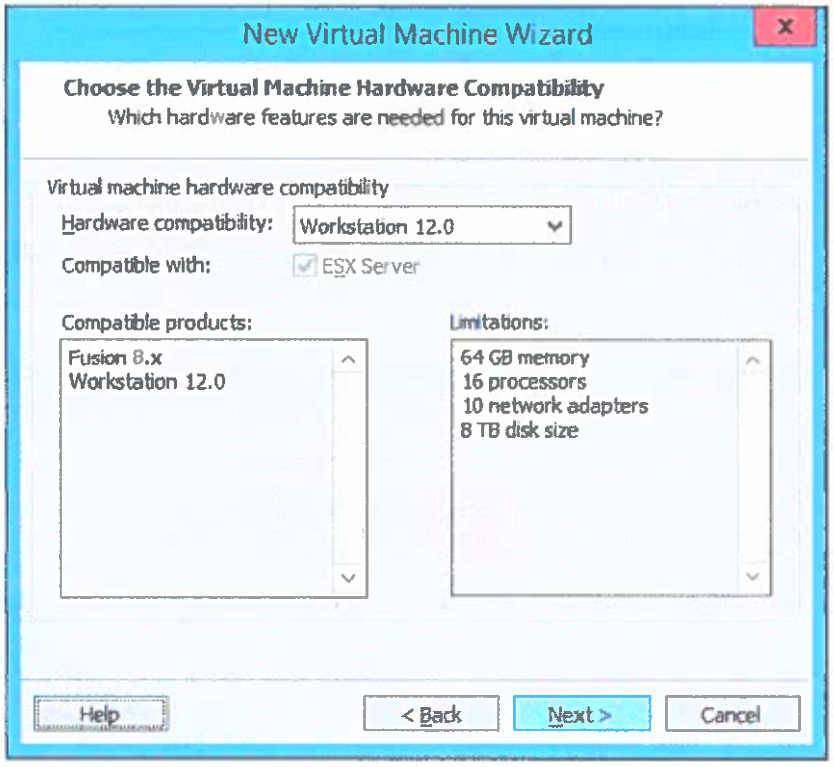

Figura 8 - Compatibilidade para o VMware WS 12

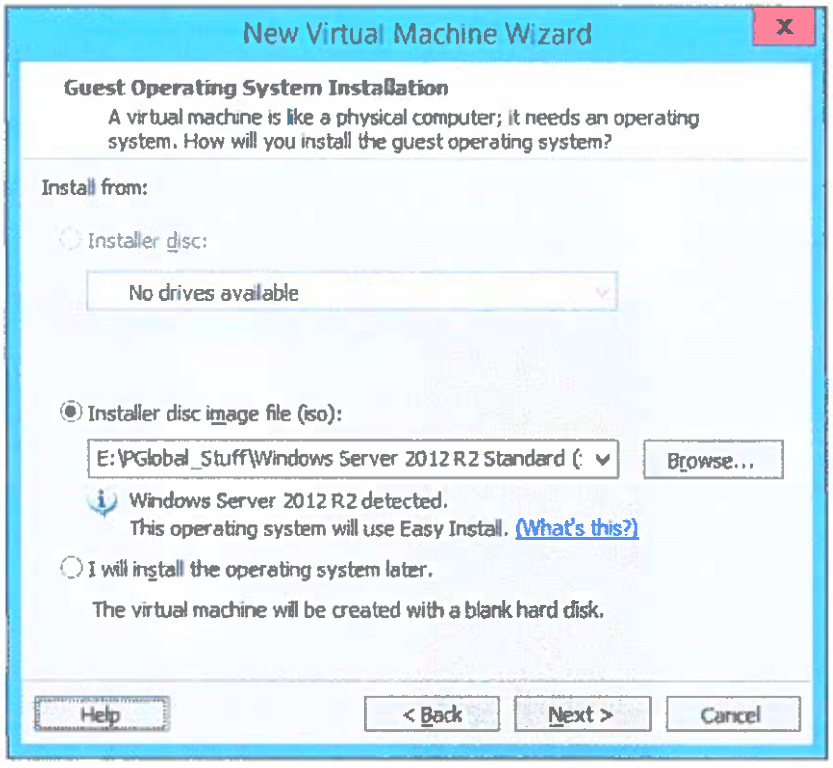

Figura 9 - Instalação através de uma imagem de disco do tipo ISO[16]

[16] Uma imagem ISO é um sistema de ficheiros no formato de CD, DVD ou BD

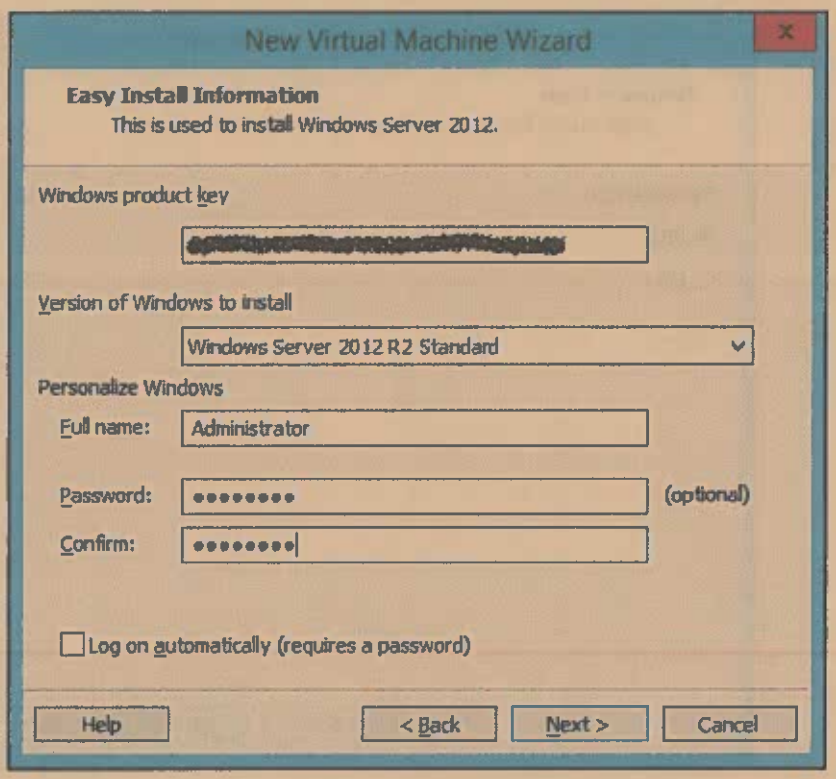

Figura 10 - Inserção da chave do produto, a versão e as credenciais

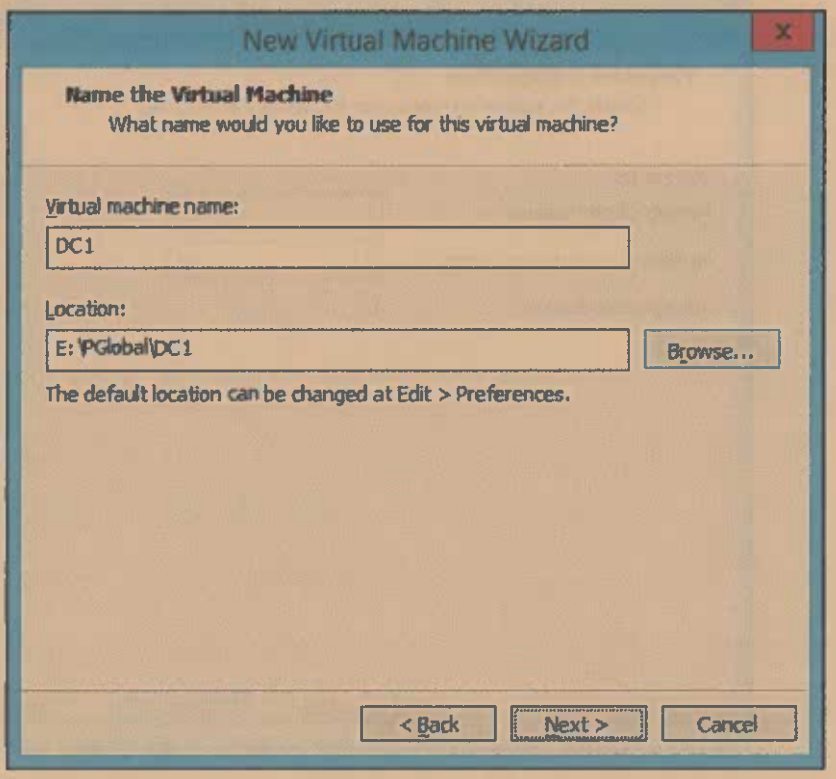

Figura 11 - Atribuição do nome e da localização do controlador de domínio

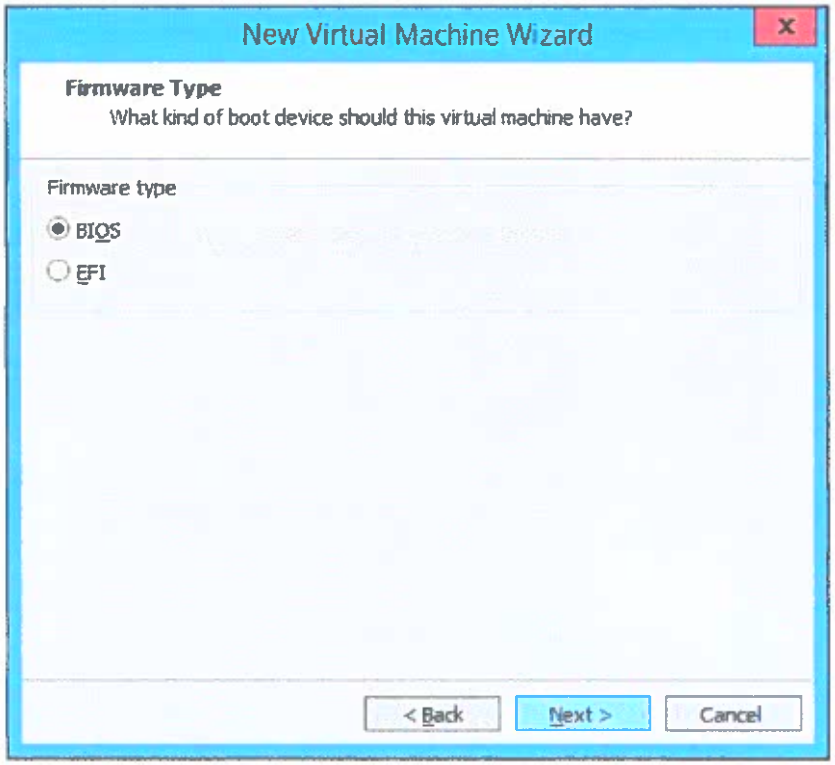

Figura 12 - Tipo de Firmware[17]

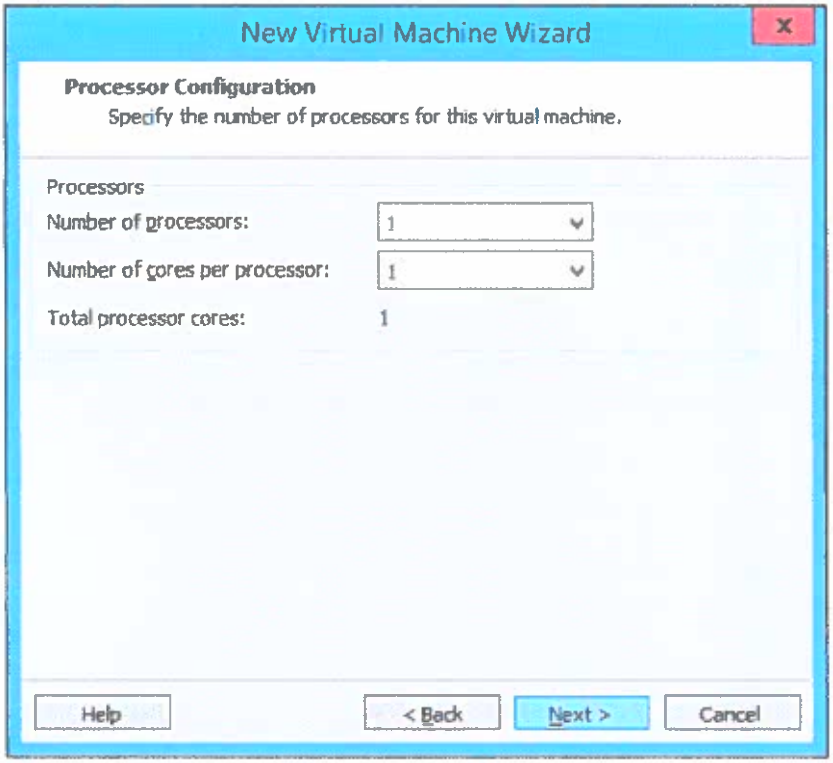

Figura 13 - Quantidade de processadores e quantos cores por cada

[17] Conjunto de instruções operacionais programadas direta e permanentemente no hardware

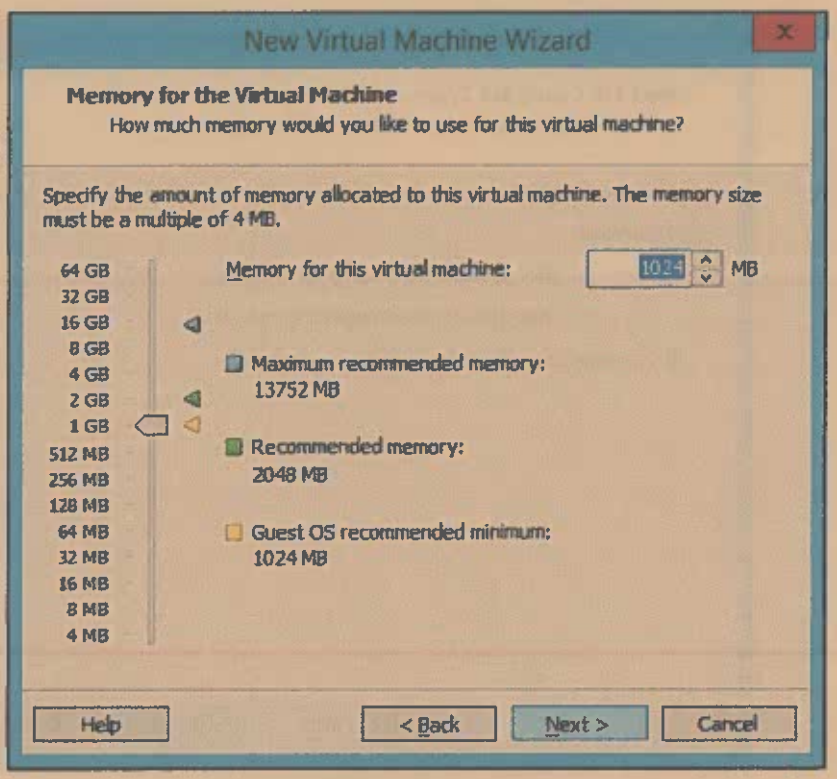

Figura 14 - Atribuição de memória

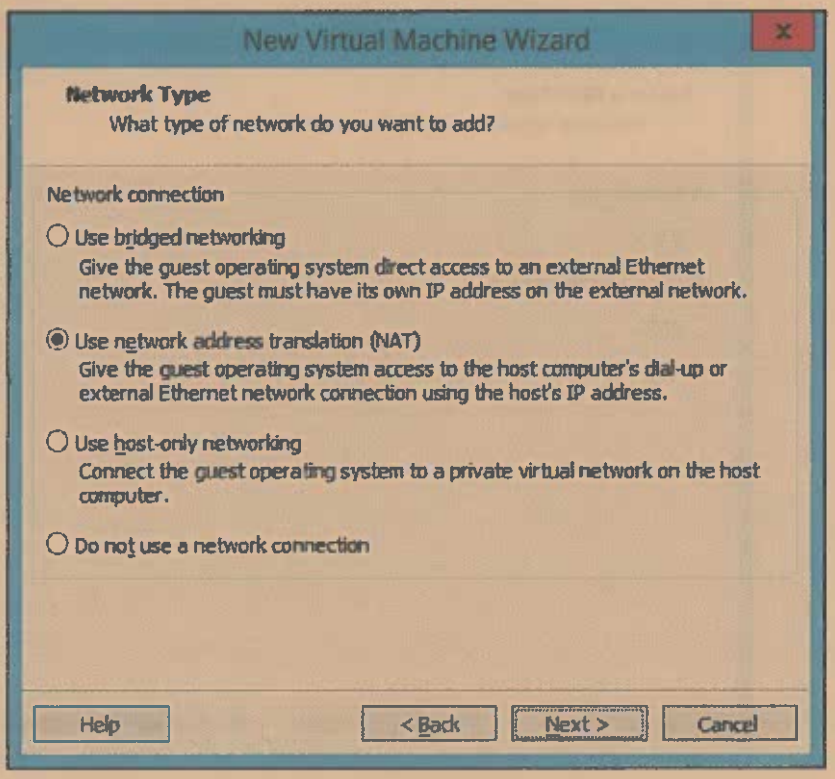

Figura 15 - Placa de rede em modo NAT para permitir acesso à rede externa

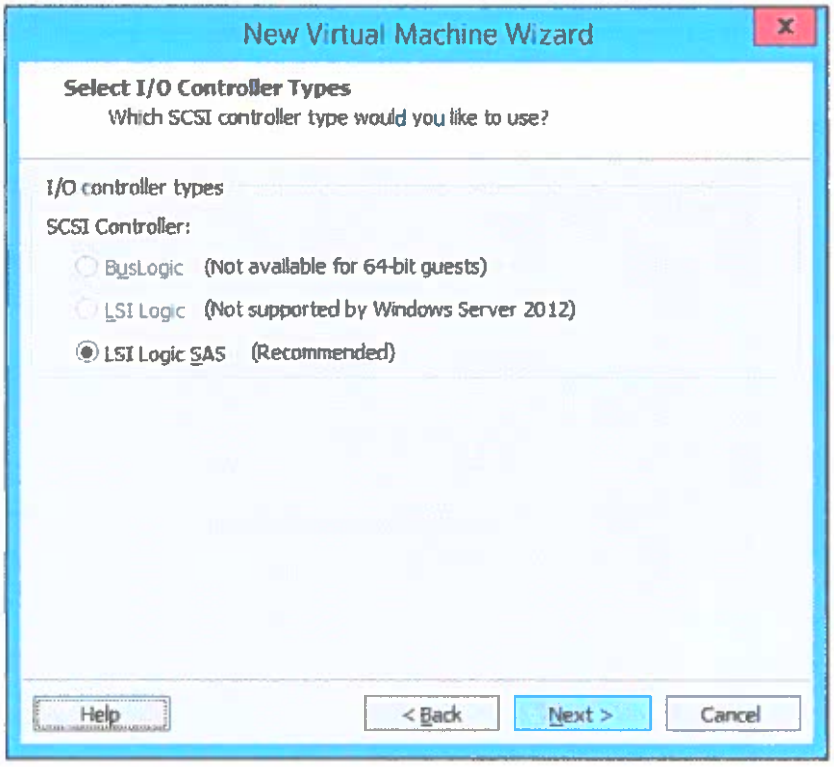

Figura 16 - Controlador SCSI

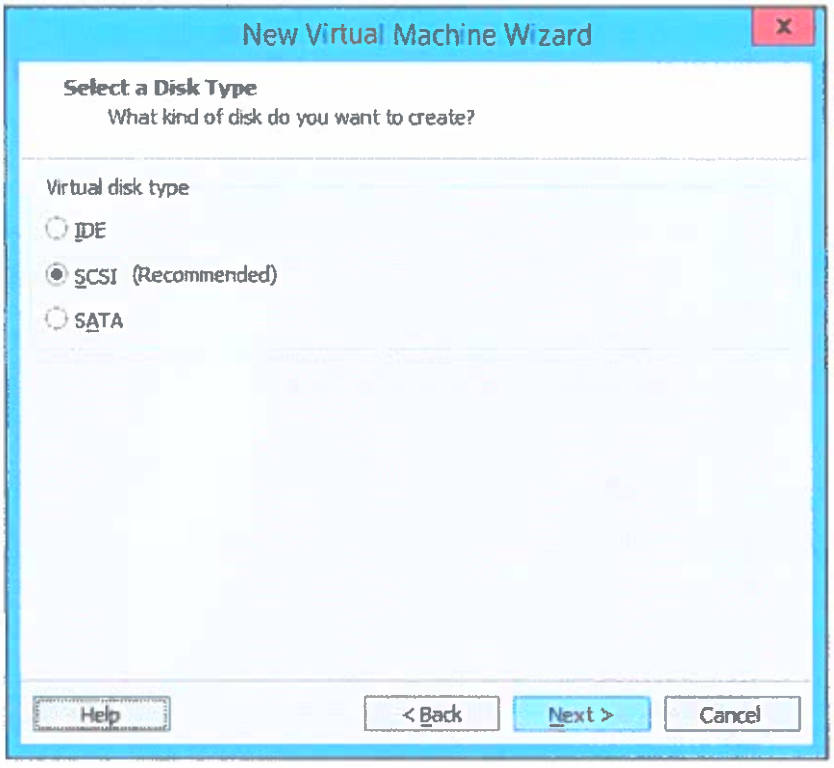

Figura 17 - Definir o tipo de disco

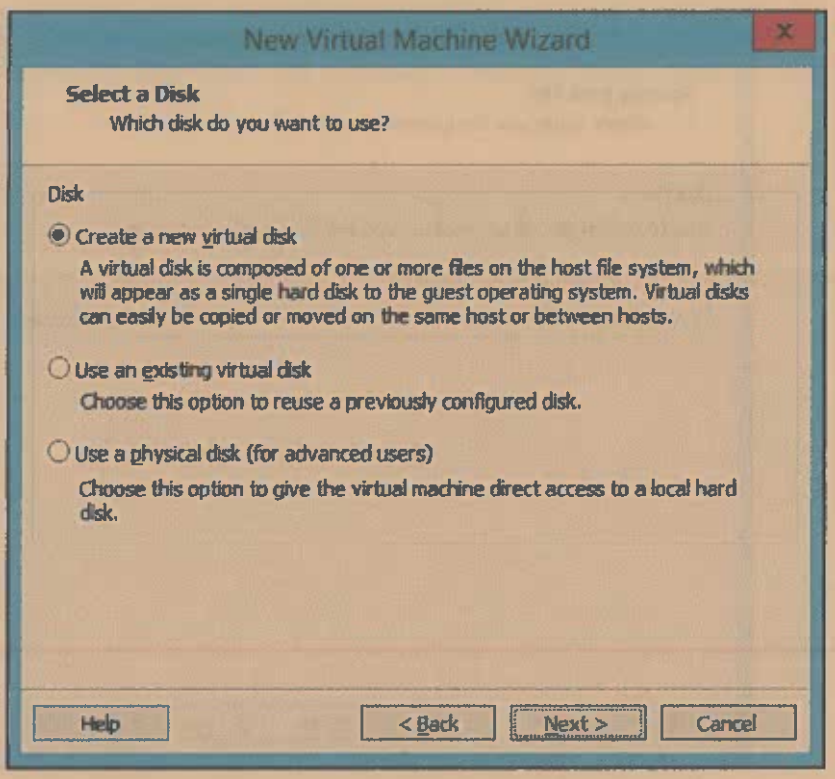

Figura 18 - Criar a máquina virtual a partir de um novo disco

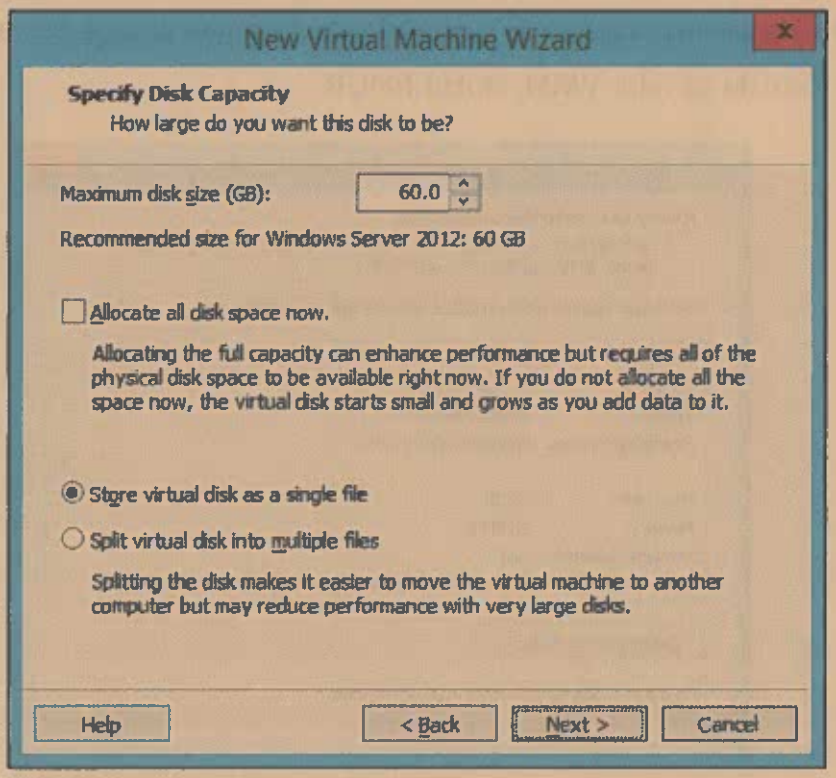

Figura 19 - Tamanho do disco guardado como um único ficheiro

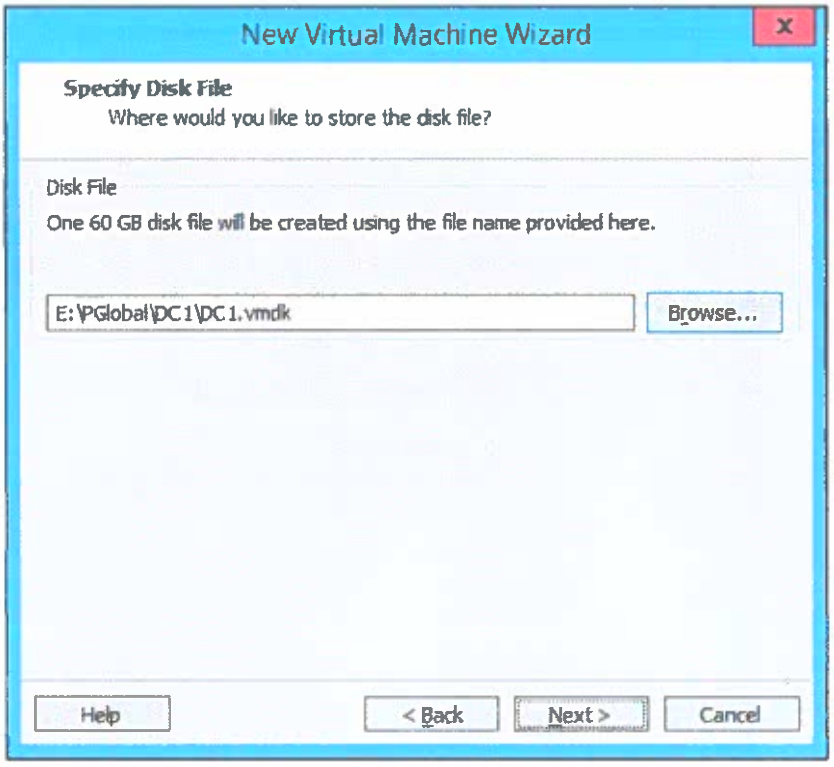

Figura 20 - Localização do disco virtual

Para o tamanho dos discos do controlador de domínio e do SQL Server atribuí 60GB e, para o disco do servidor VMM, atribuí 100GB.

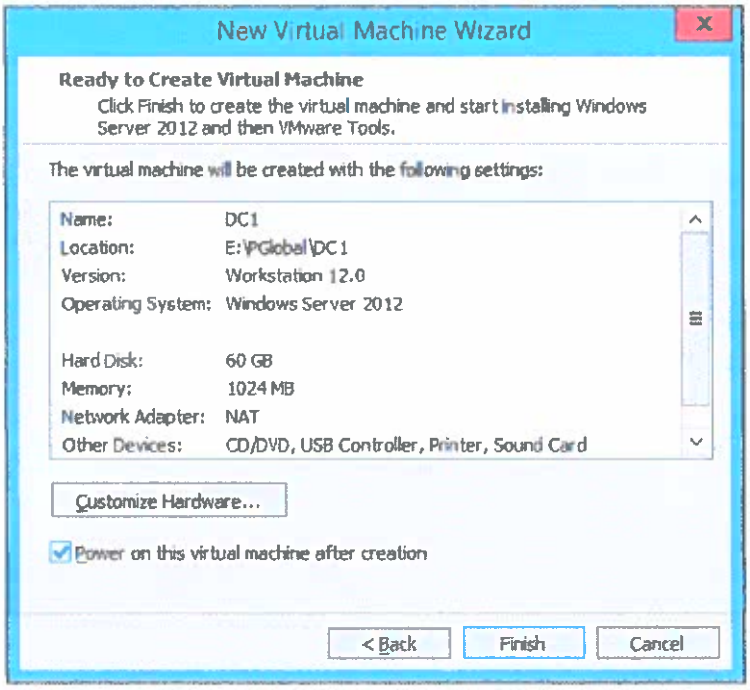

Figura 21 - Resumo das configurações da máquina DC1

Criada a máquina virtual, a mesma inicia e instala-se o Windows Server 2012 R2 Standard Edition. Após as atualizações do *Windows Update*, promove-se o servidor a controlador de domínio. Para isso deve-se clicar em "Add roles and features", na consola do Server Manager (figura 20).

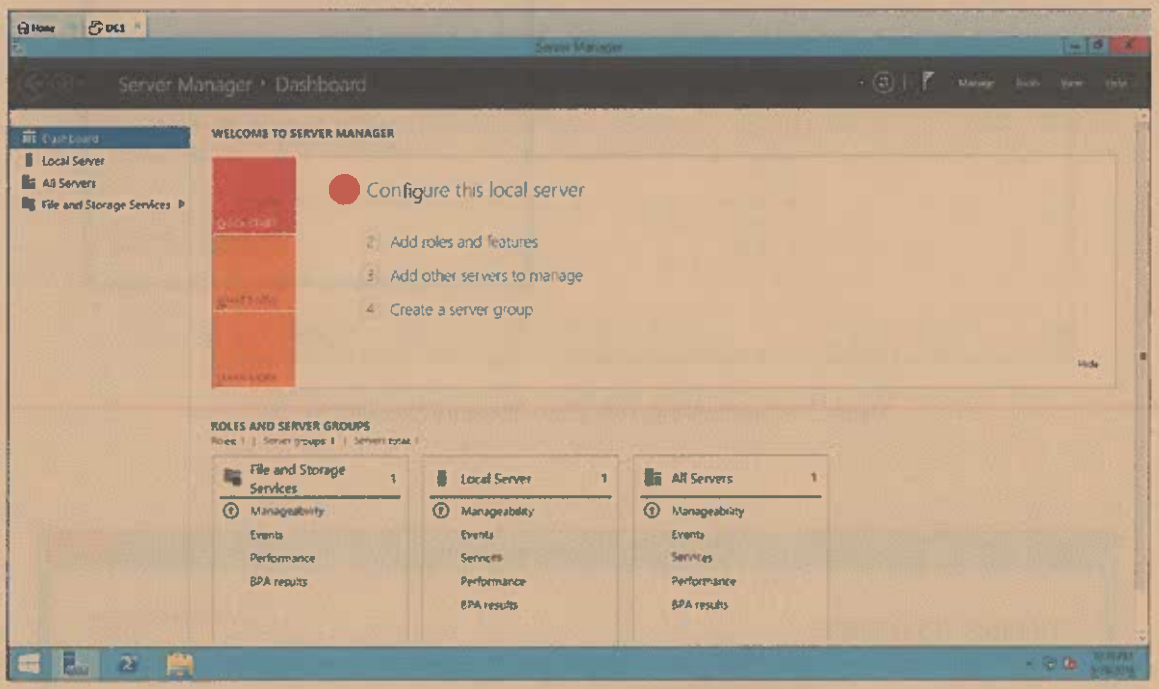

Figura 22 - Consola para a gestão do servidor DC1

Na janela "Add Roles and Features Wizard", escolher "Role-based or feature-based installation" como tipo de instalação, escolhe-se o servidor DC1 e a role "Active Directory Domain Services". Ao escolher essa role, aparece uma janela para a instalação das "Remote Server Administration Tools".

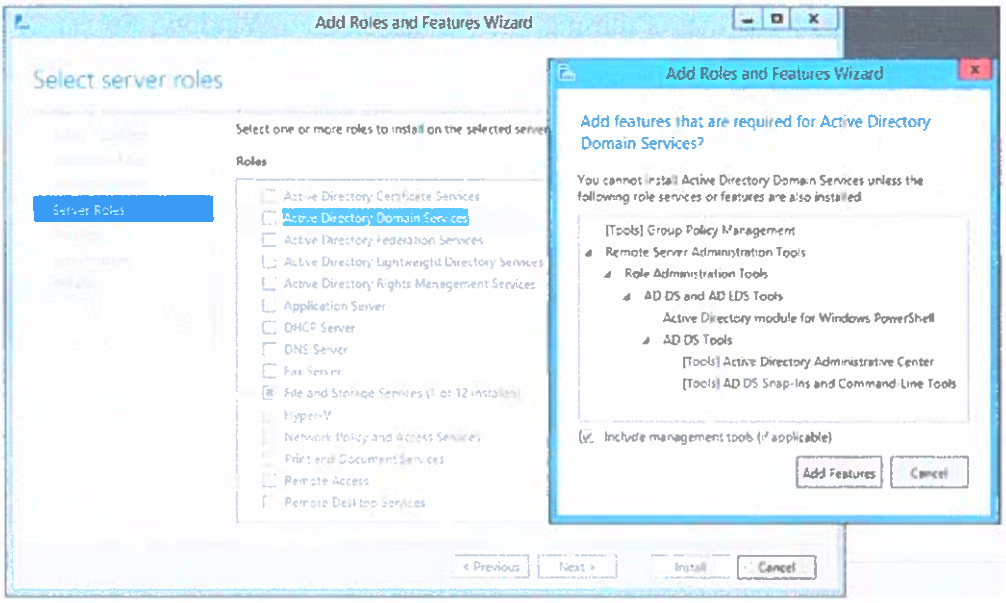

Figura 23 - Instalação da role Active Directory Domain Services

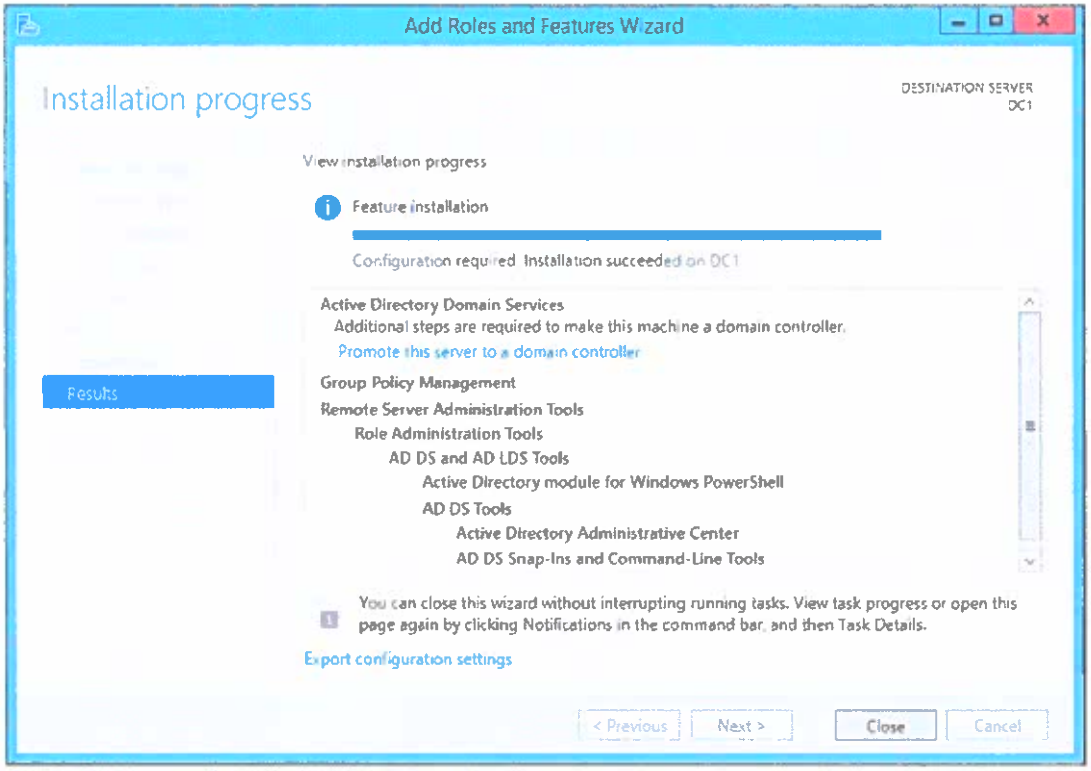

Figura 24 - Finalização da instalação da role

Para promover o servidor a controlador de domínio cliquei no link "Promote this server to a domain controller".

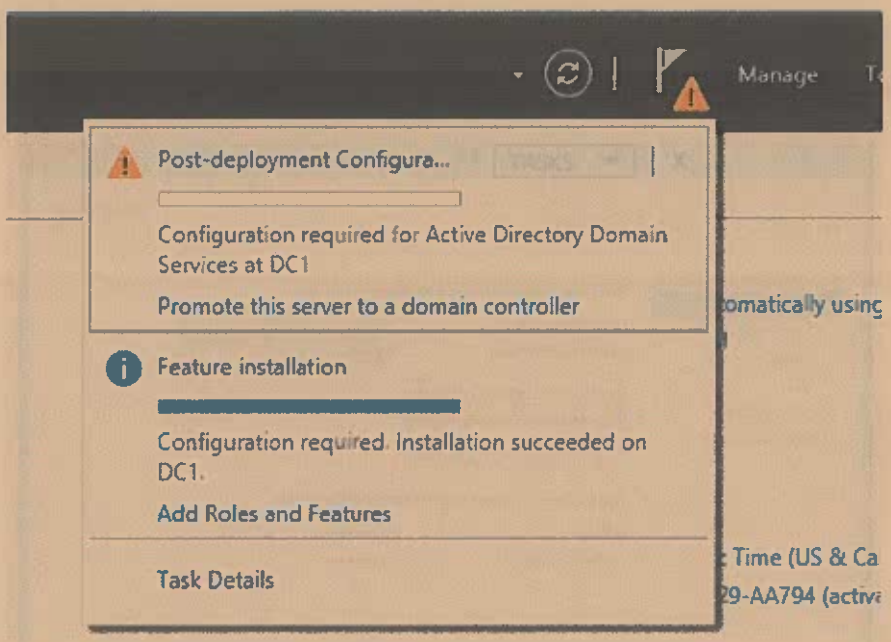

Figura 25 - Janela de alerta para promoção do servidor

No separador "Deployment Configuration", escolher a opção "Add a new forest" e especificar o nome do domínio "DigitalLib.local".

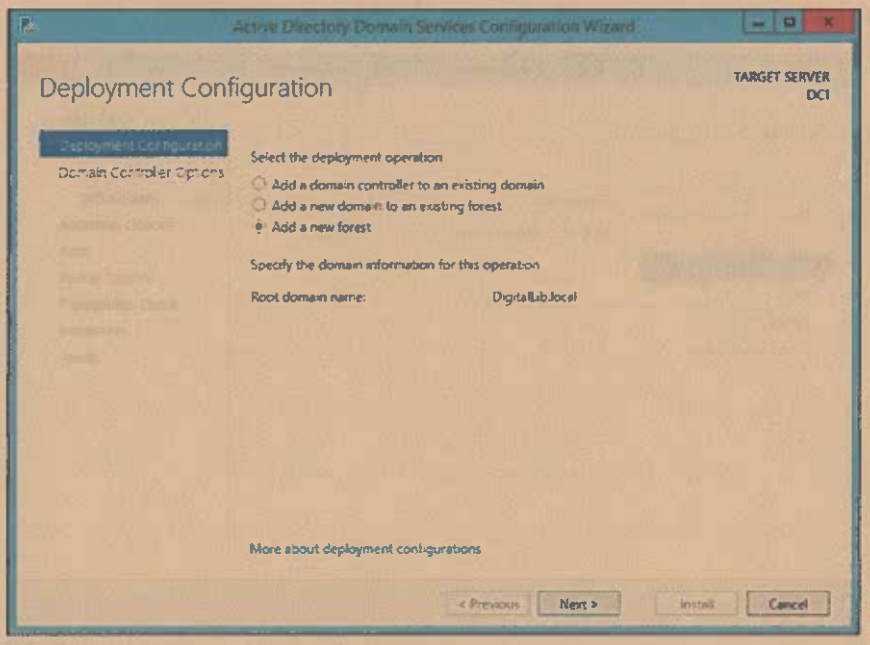

Figura 26 - Adicionar uma nova floresta

No separador "Domain Controller Options" deixar os níveis de funcionalidade com os valores "Windows Server 2012 R2" e colocar a *password*.

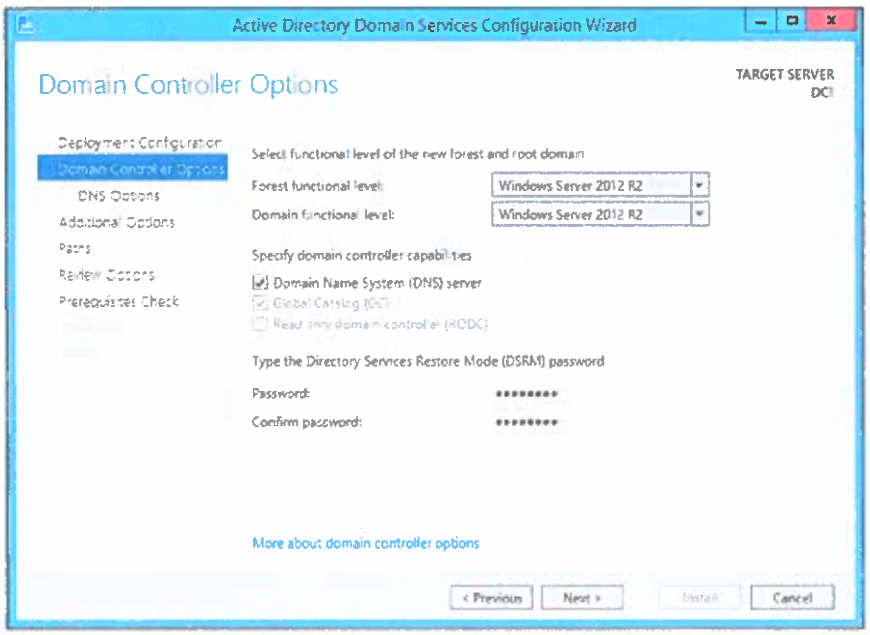

Figura 27 - Nível funcional da floresta e do domínio principal

Na próxima figura é mostrado o nome do domínio NetBIOS.

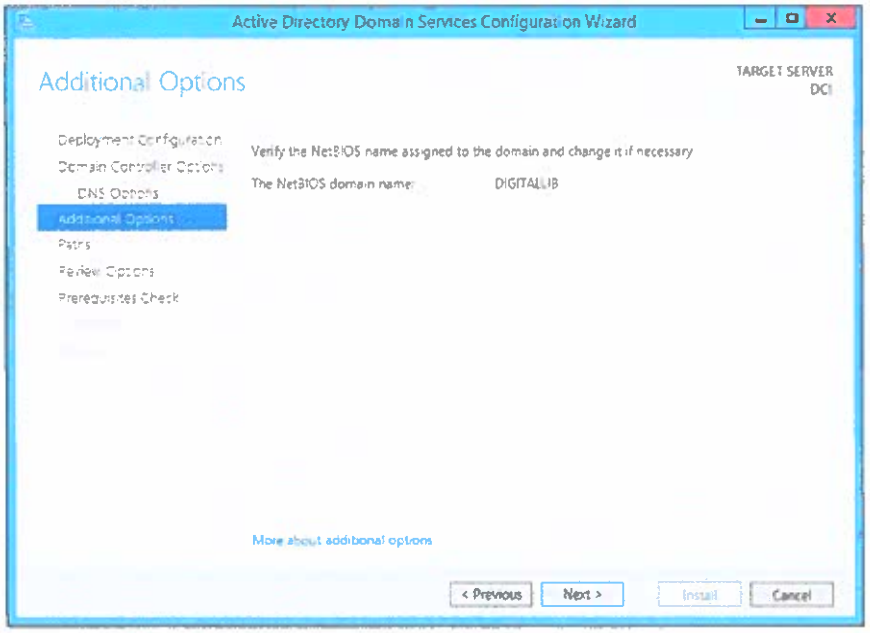

Figura 28 - Verificação do nome do domínio NetBIOS

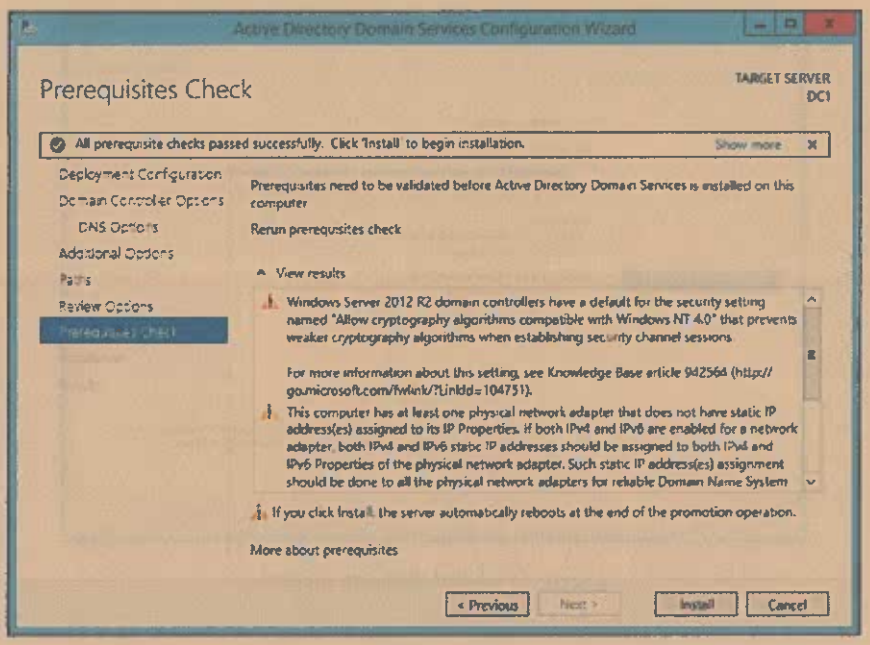

Figura 29 - Verificação dos requisitos para a promoção a controlador de domínio

Após o término da configuração do Active Directory Domain Services, reinicia-se a máquina virtual. Aguarda-se que a consola Server Manager abra e verifica-se que o servidor já se encontra promovido a controlador do domínio "DigitalLib.local" (figura 28).

b) Instalação do servidor de DHCP

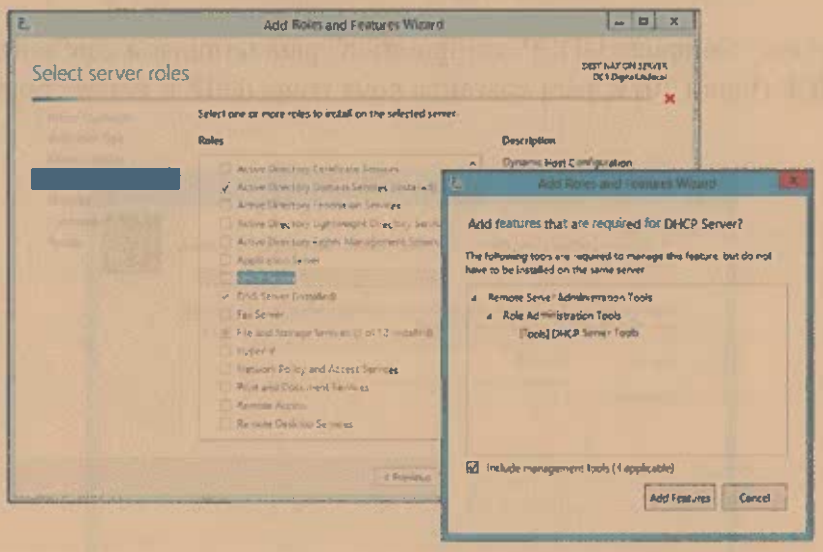

Figura 30 - Instalação da role "DHCP Server" e as ferramentas administrativas

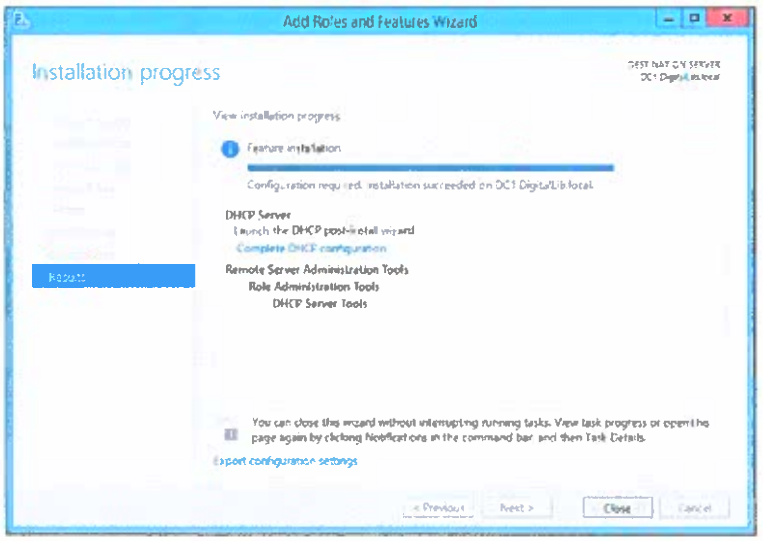

Figura 31 - Instalação com sucesso

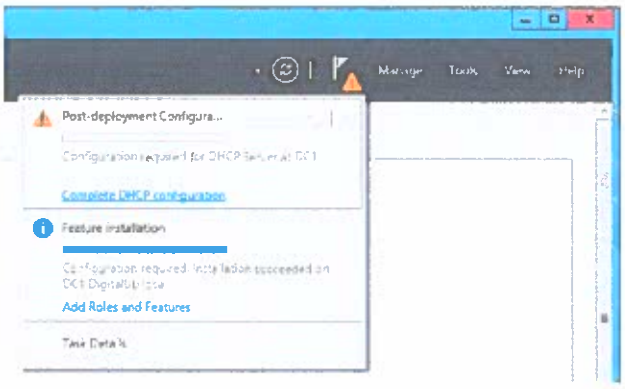

Figura 32 - Alerta para terminar a configuração do servidor DHCP

Cliquei em "Complete DHCP configuration" para terminar a configuração do servidor DHCP (figura 30) e, para criar uma nova range de IP's, comecei por dar um nome.

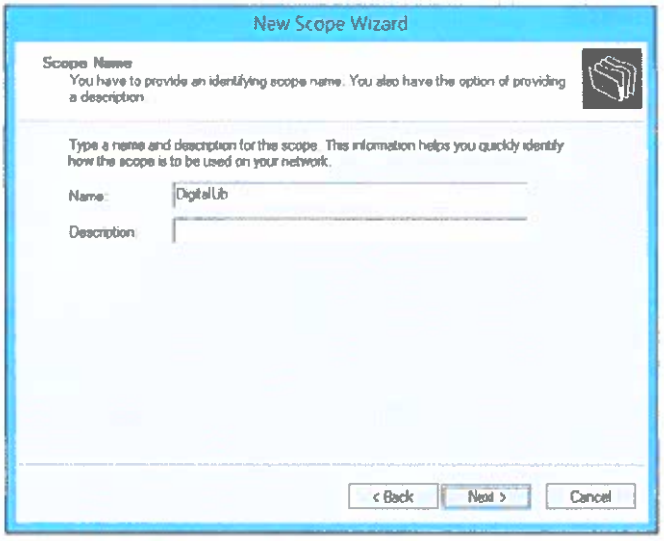

Figura 33 - Assistente de criação de range de IP's

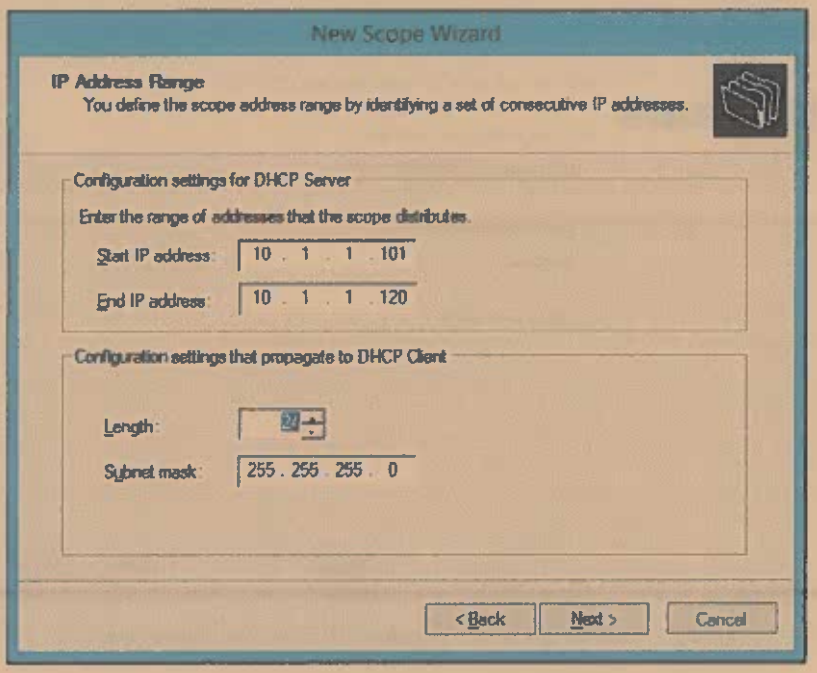

Figura 34 - Configuração dos IP's

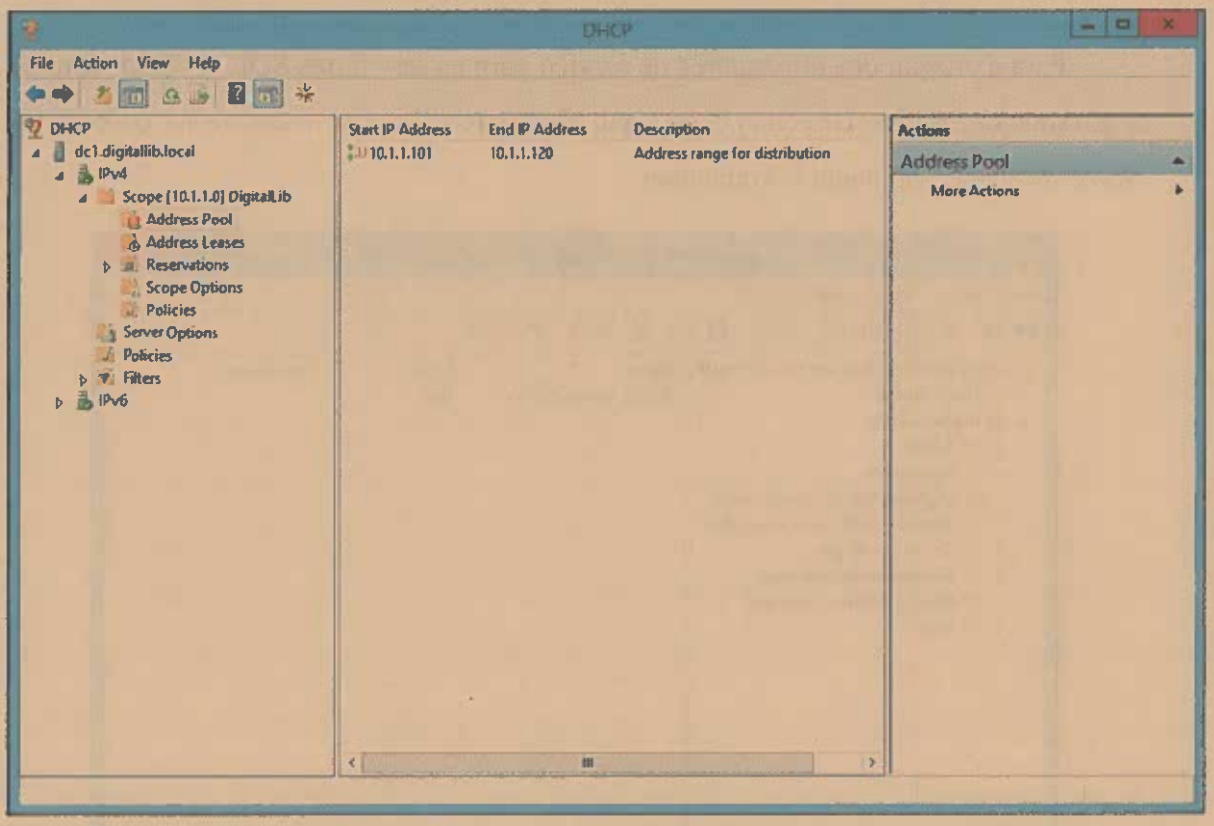

Figura 35 - Intervalo de IP's para distribuição

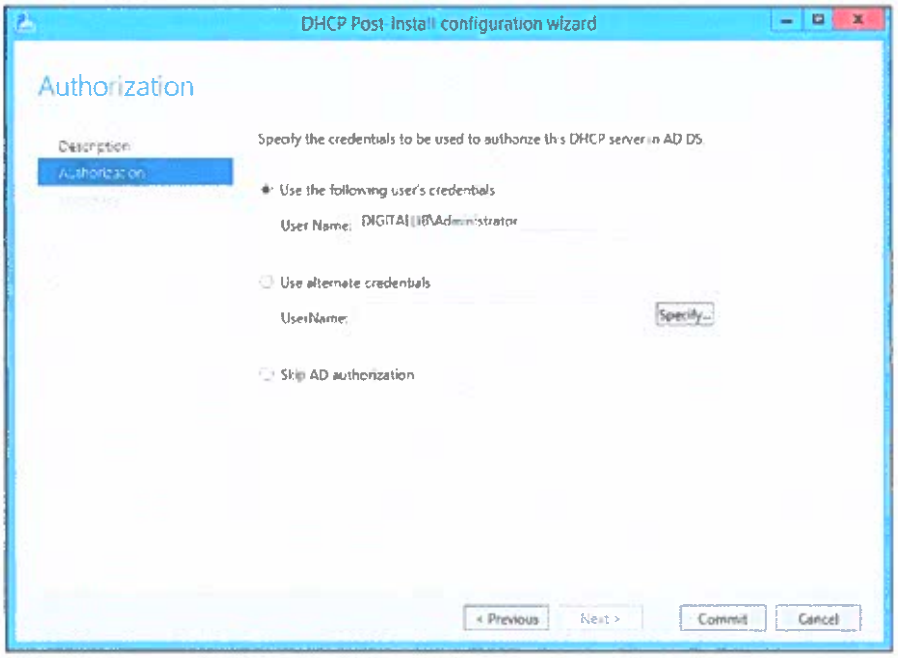

Figura 36 - Credenciais para o servidor DHCP usar na Active Directory

c) Criação das unidades organizacionais

Para a criação dos utilizadores de serviço para os servidores SQL e VMM, abre-se a ferramenta "Active Directory Users and Computers" que se encontra na consola do Server Manager, no menu Ferramentas.

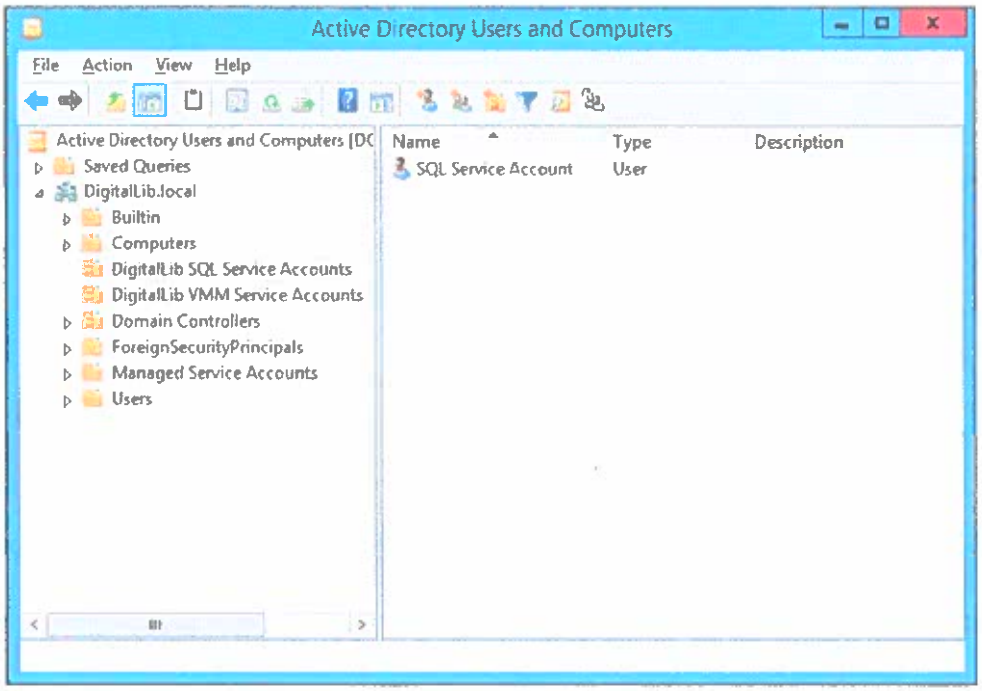

Figura 37 - Utilizador de serviço para o SQL Server

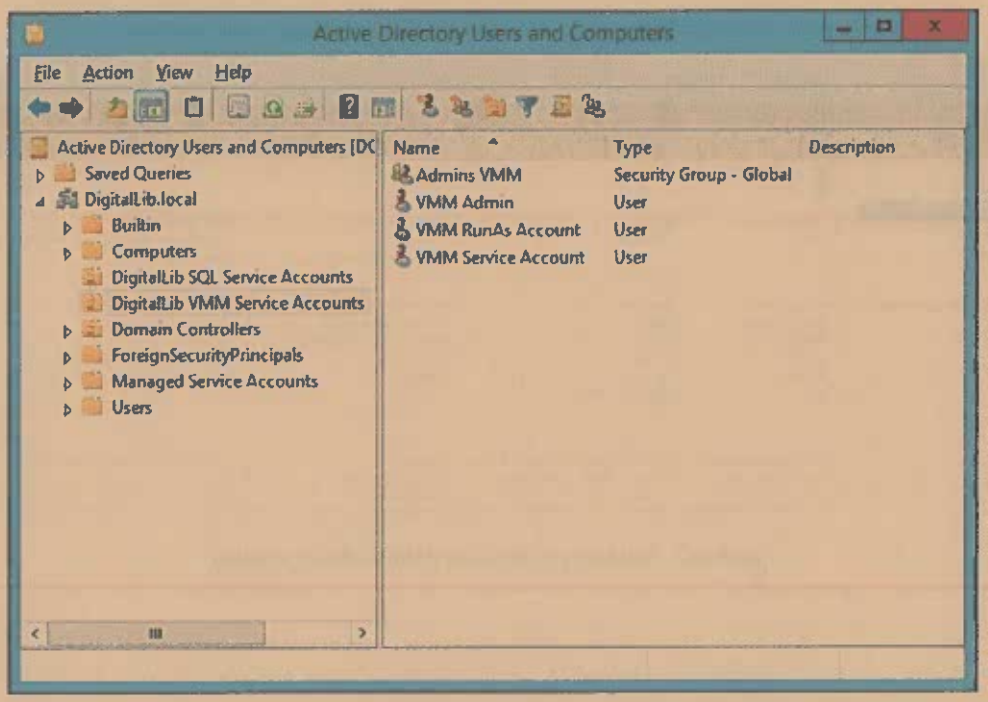

Figura 38 - Utilizadores de serviço para o System Center Virtual Machine Manager

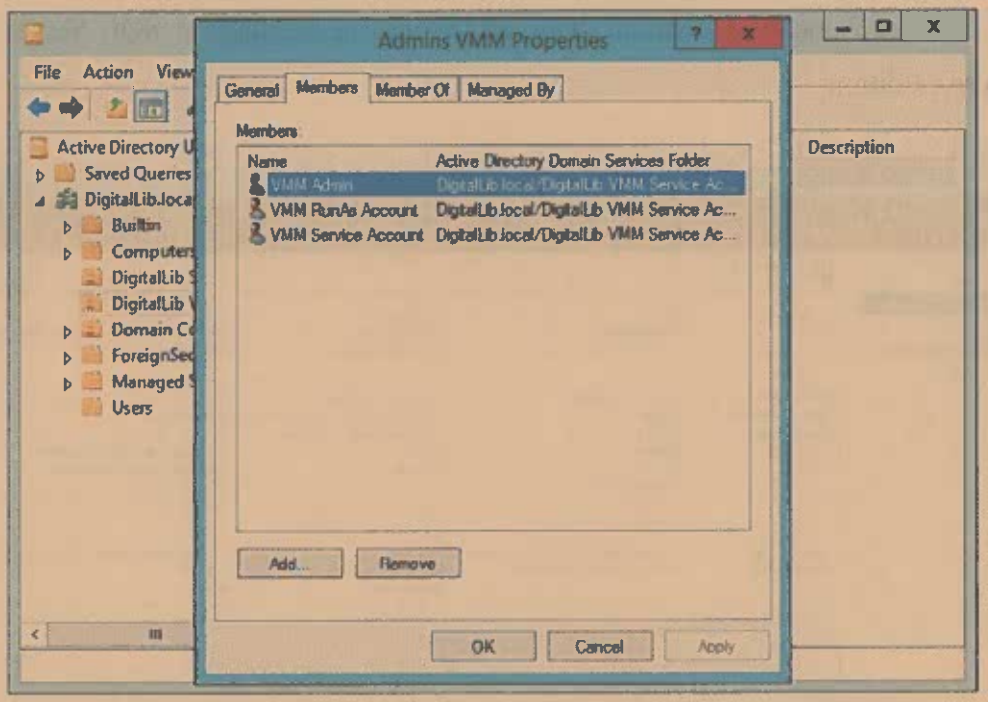

Figura 39 - Grupo de administração dos utilizadores de serviço

d) SQL – Servidor de base de dados

Figura 40 - Servidor promovido a controlador de domínio

Após criar o controlador de dominio, foi criada outra máquina com o nome SQL com o Windows Server 2012 R2 Datacenter Edition para ser um servidor de base de dados. Foi adicionado ao dominio "DigitalLib.local" e instalado o SQL Server 2012 Enterprise Edition.

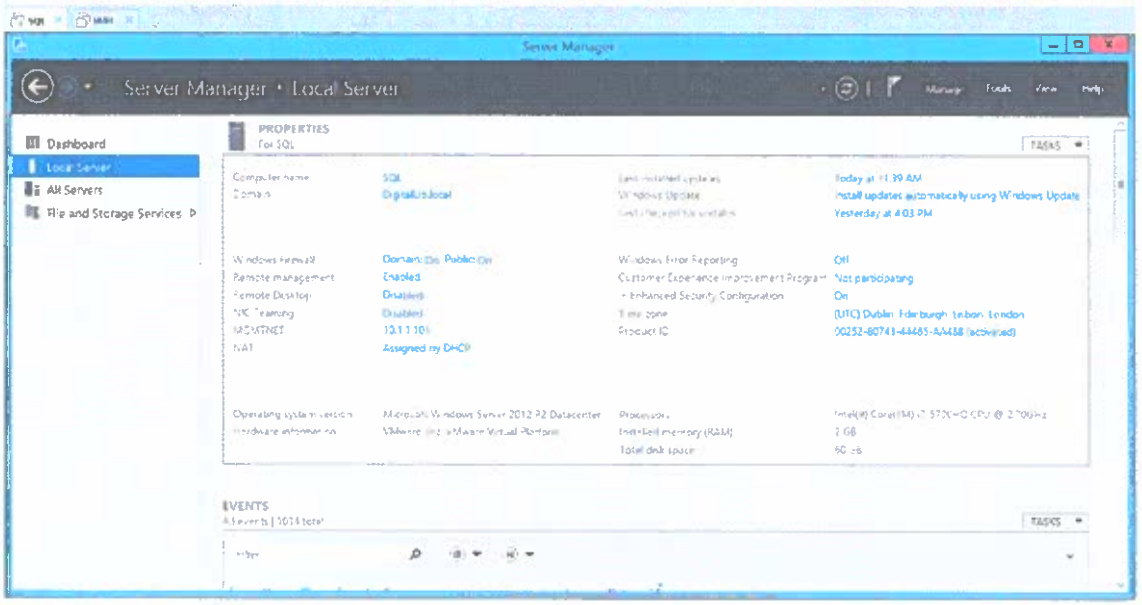

Figura 41 - Servidor para o SQL Server

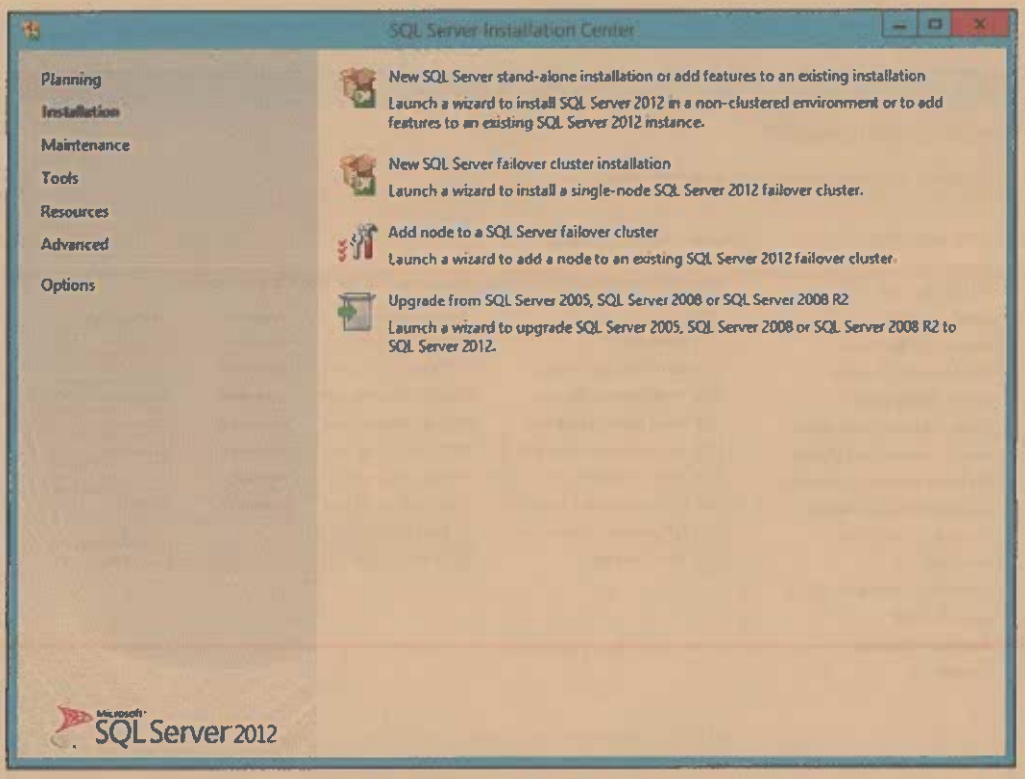

Figura 42 - Instalação do SQL Server 2012 Enterprise Edition

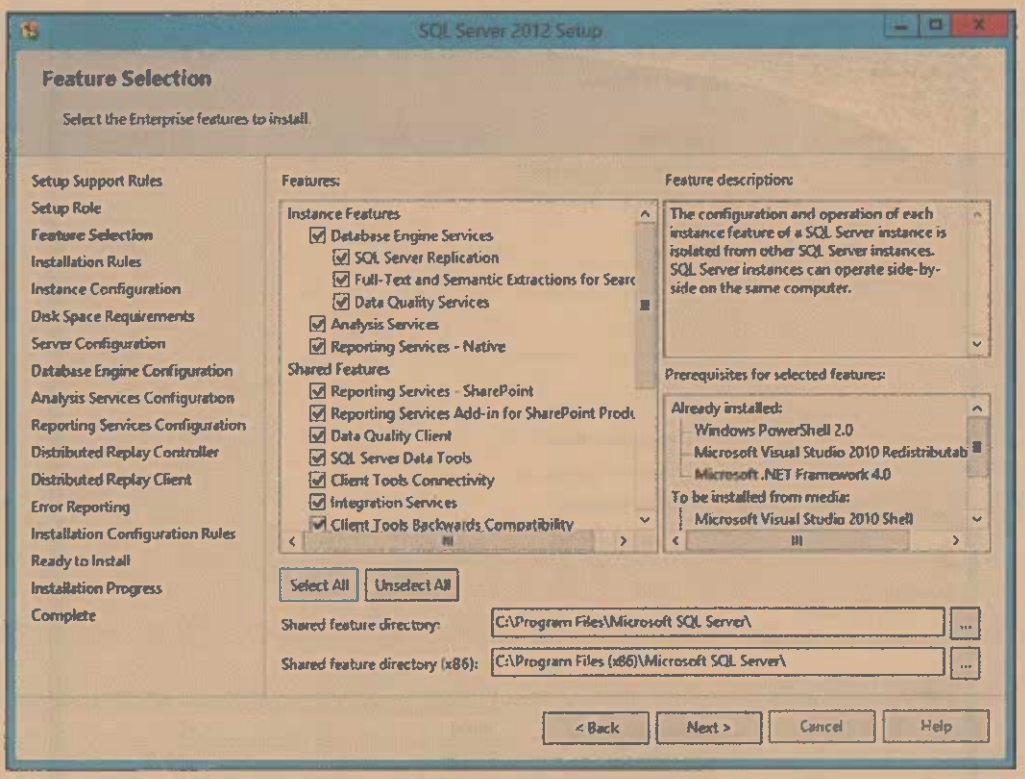

Figura 43 - Seleção completa dos componentes a instalar

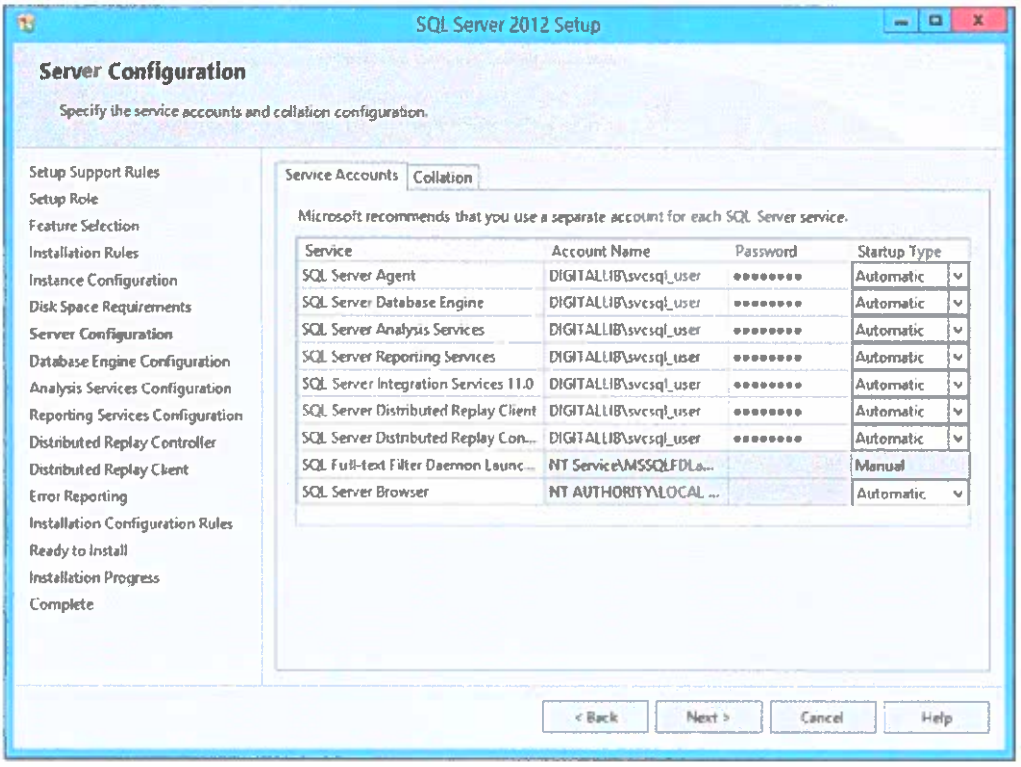

Figura 44 - Credenciais para uso nos serviços do SQL Server

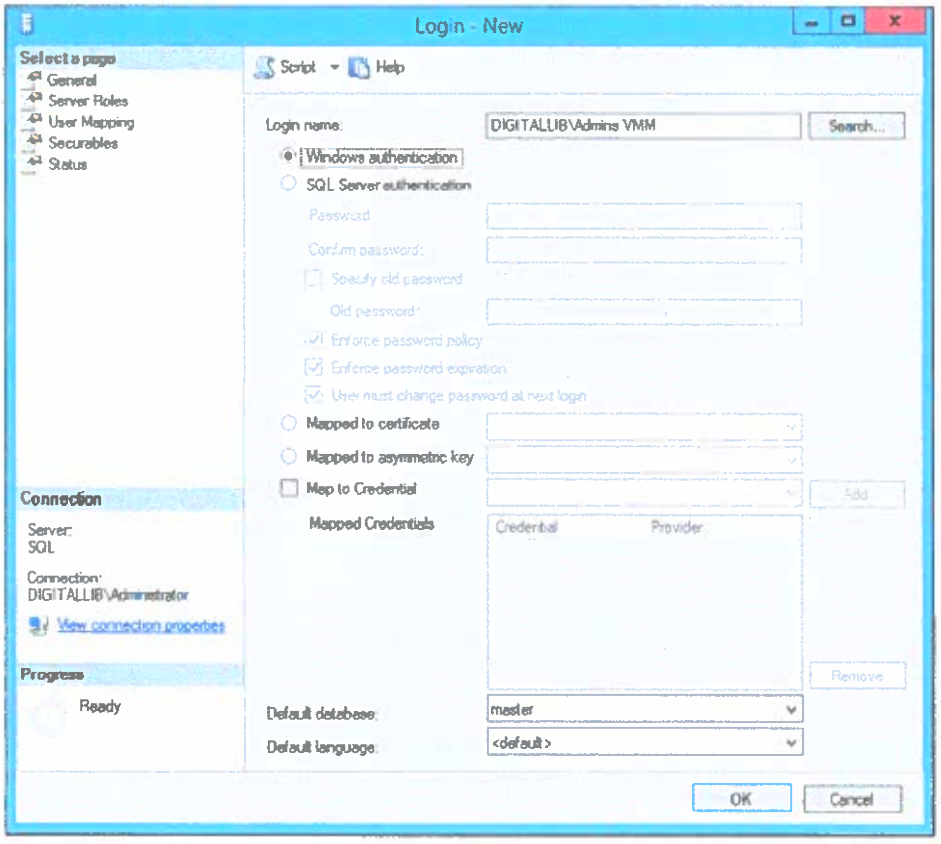

Figura 45 - Criar um login do grupo de administrador do VMM

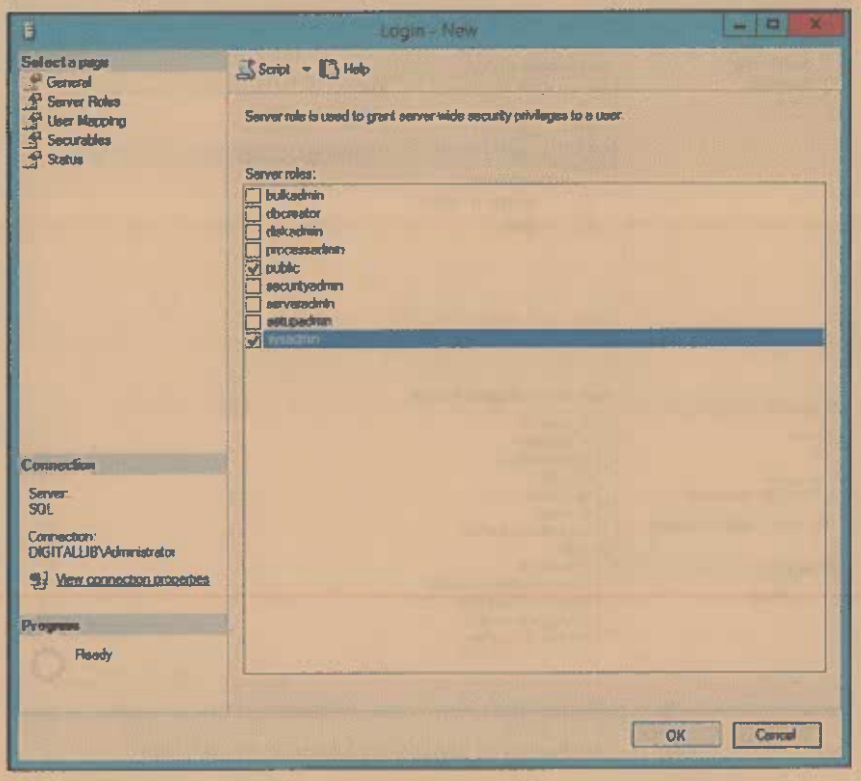

Figura 46 - Atribuição de privilégios ao grupo de administradores do VMM

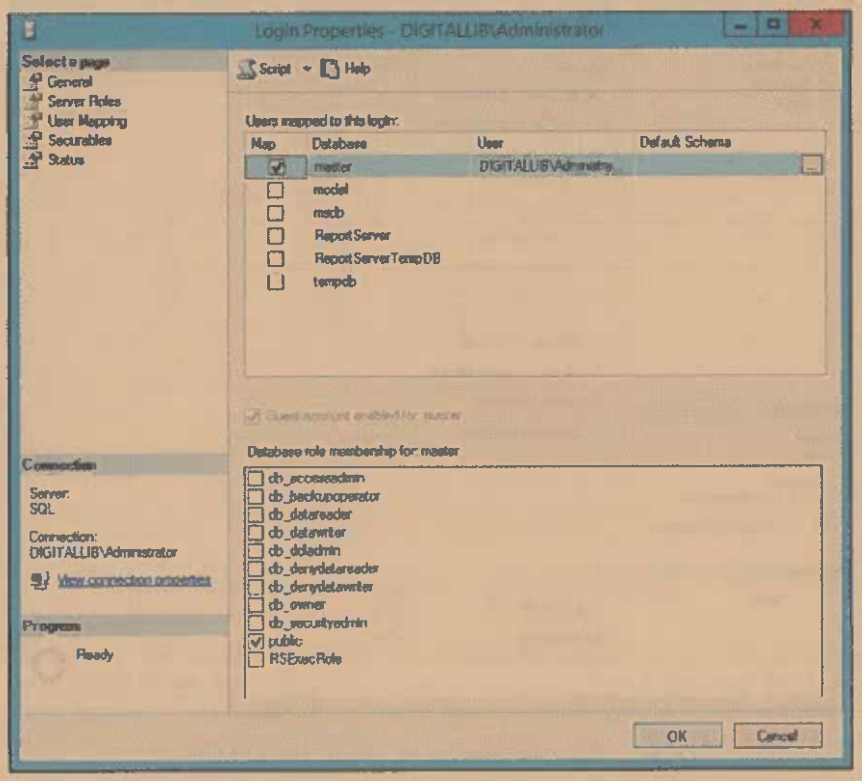

Figura 47 - Atribuição de permissões à base de dados 'master'

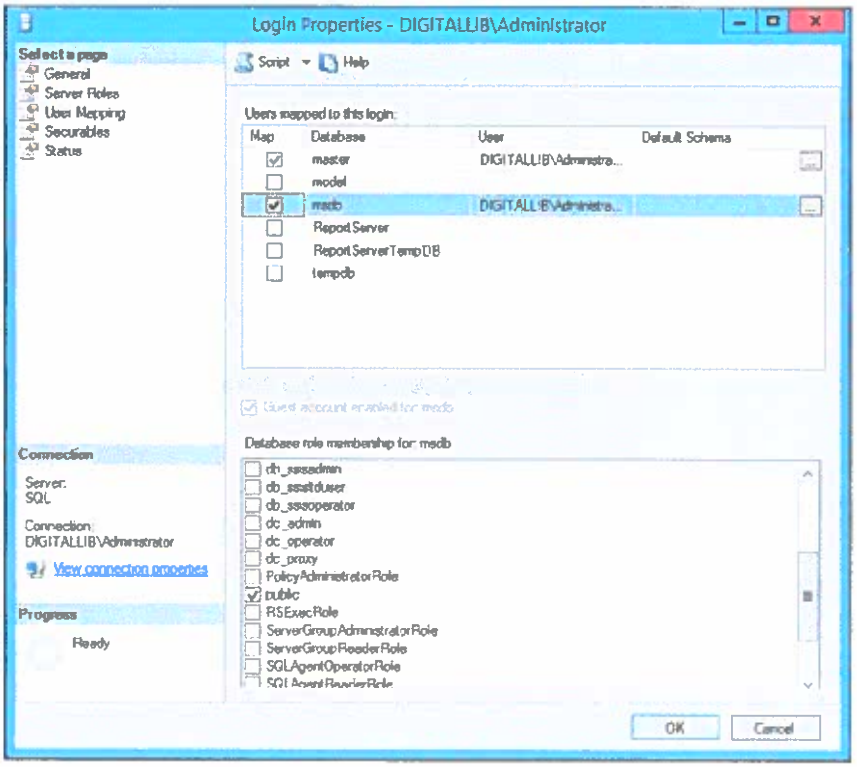

Figura 48 - Atribuição de permissões à base de dados 'msdb'

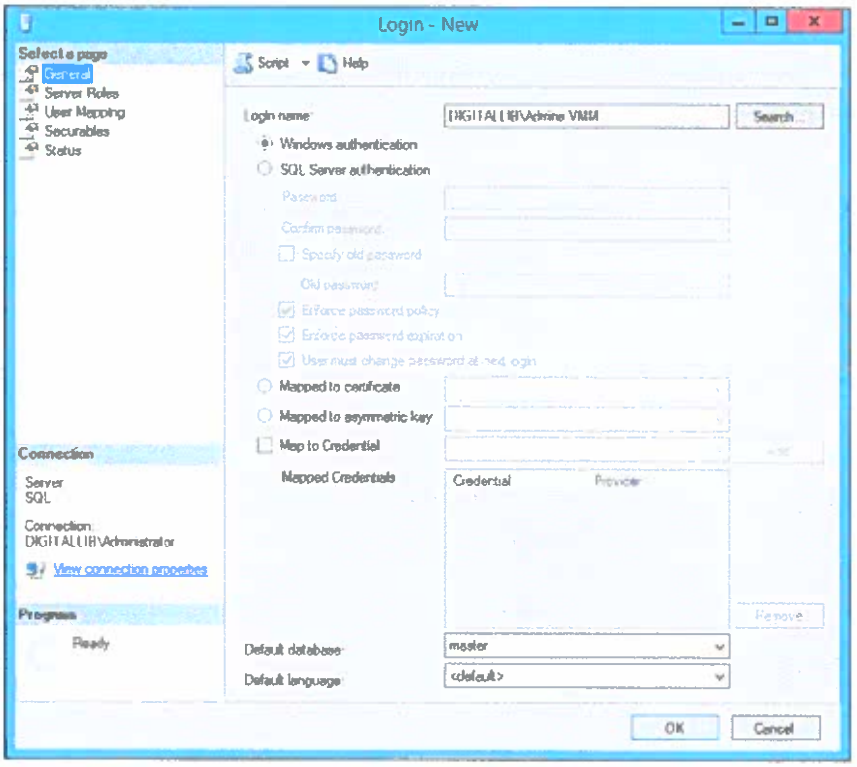

Figura 49 - Novo login para o grupo 'Admins VMM'

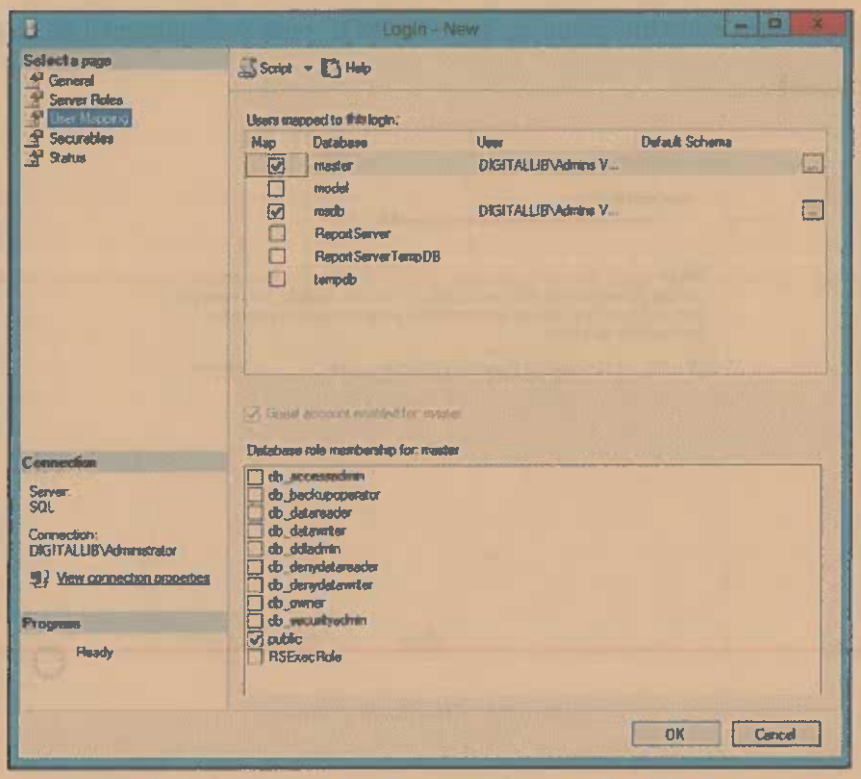

Figura 50 - Permissões para o grupo 'Admins VMM' para as BD's 'master' e 'msdb'

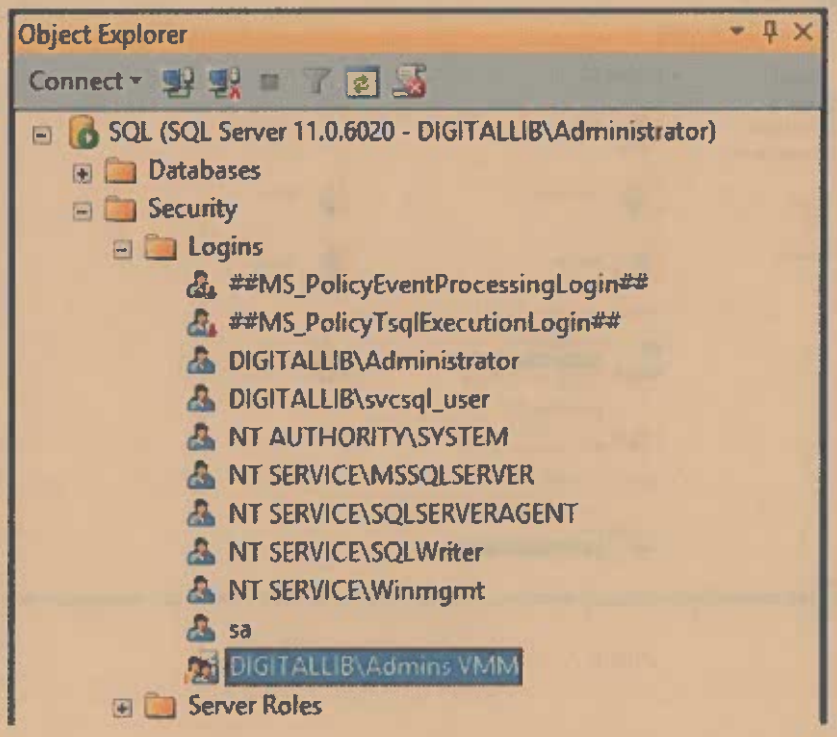

Figura 51 - Novo login 'Admins VMM' configurado

Foi adicionado mais um disco ao servidor SQL para o alojamento da base de dados da aplicação.

Figura 52 - Criação do 2º disco

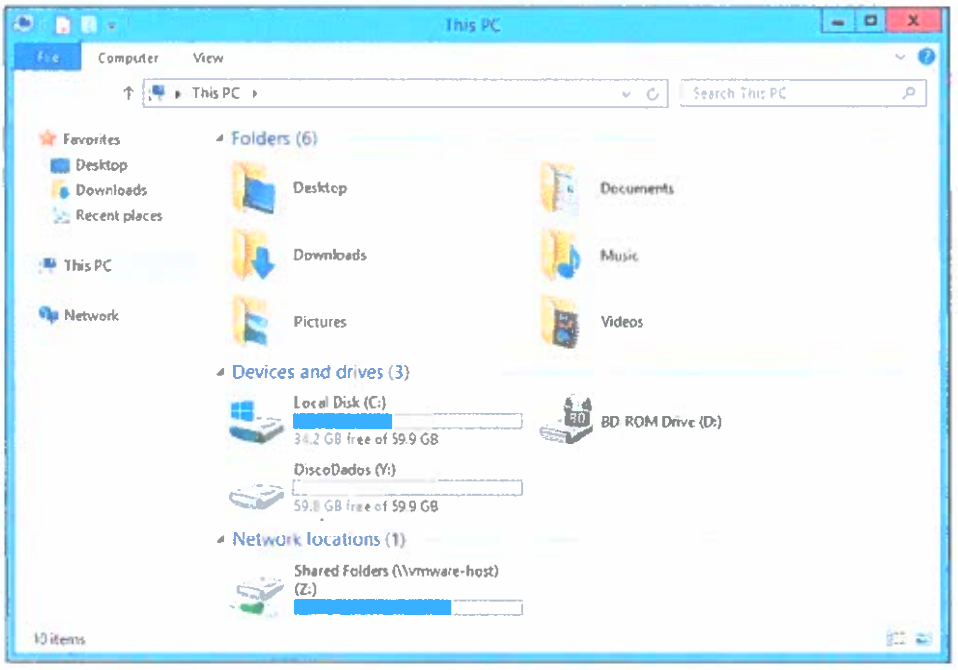

Figura 53 - Disco de dados criado e inicializado

e) VMM – Servidor de virtualização

Figura 54 - Servidor para o System Center Virtual Machine Manager

Após criar o servidor de base de dados, foi criada outra máquina com o nome VMM com o Windows Server 2012 R2 Datacenter Edition, para ser um servidor de virtualização. Foi adicionado ao domínio "DigitalLib.local" e instalado o System Center Virtual Machine Manager.

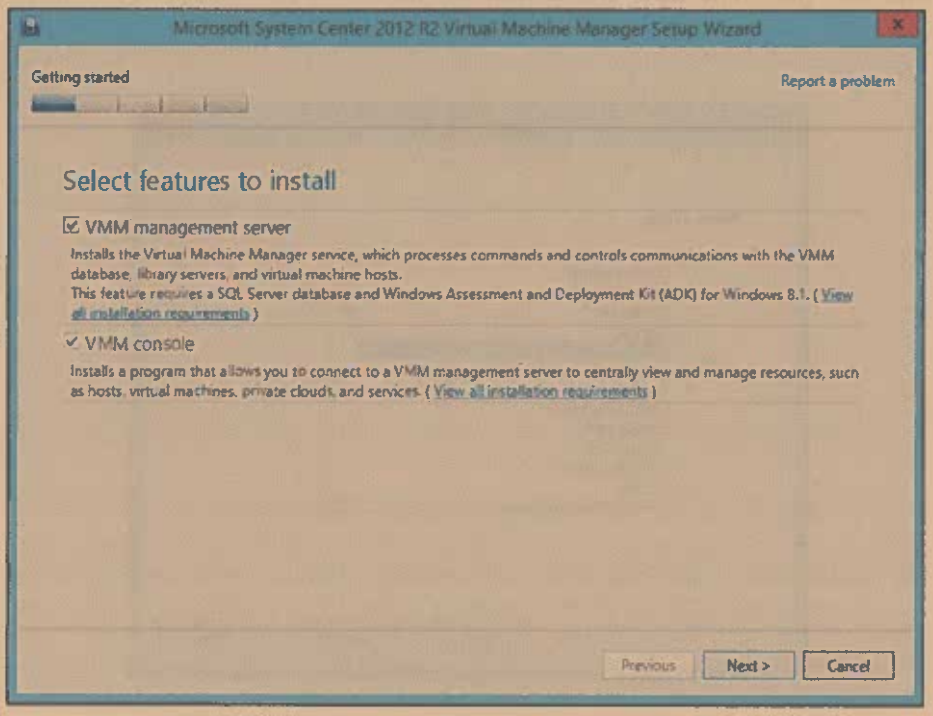

Figura 55 - Instalação do Microsoft System Center 2012 R2 Virtual Machine Manager

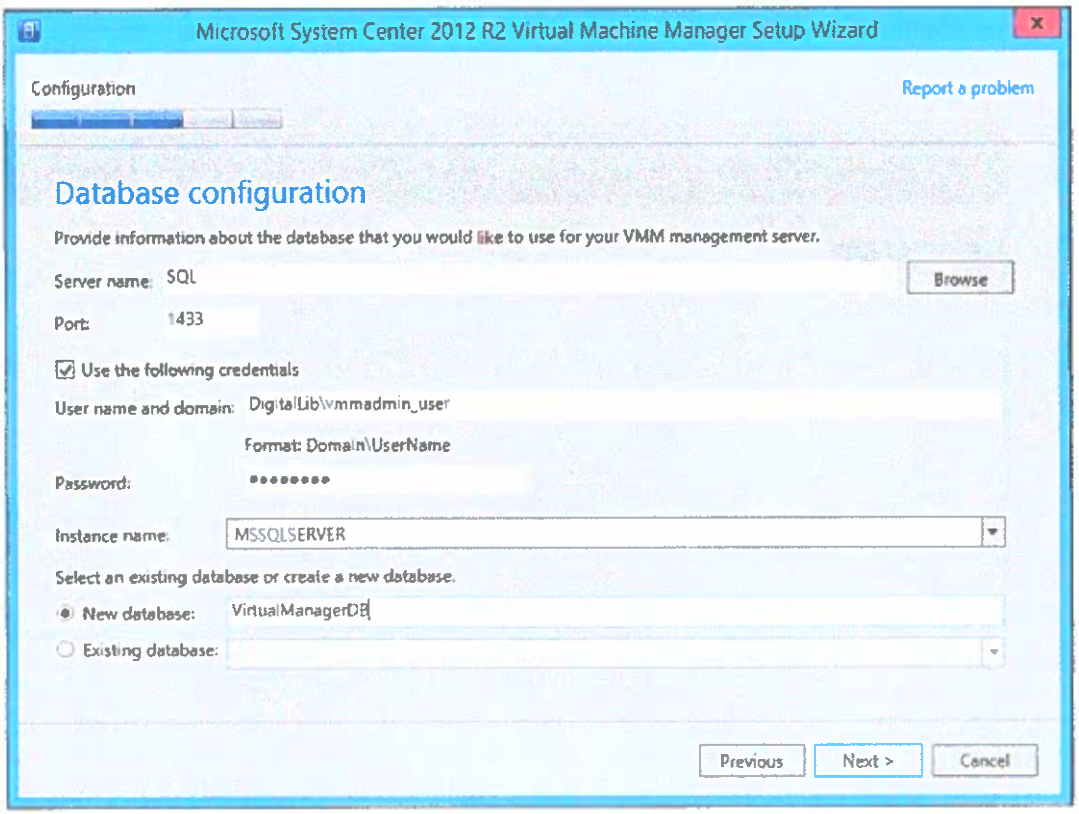

Figura 56 - Ligação ao servidor SQL Server e criação da BD do VMM

No servidor DC1 (controlador de domínio) foi criado um objeto no Active Directory Service Interface para usar uma distributed key na instalação do System Center 2012 R2 VMM.

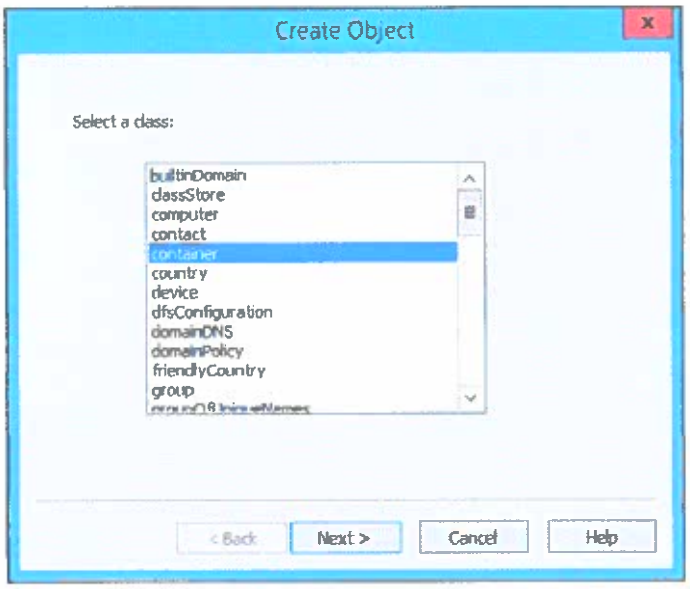

Figura 57 - Criar objeto do tipo 'Container'

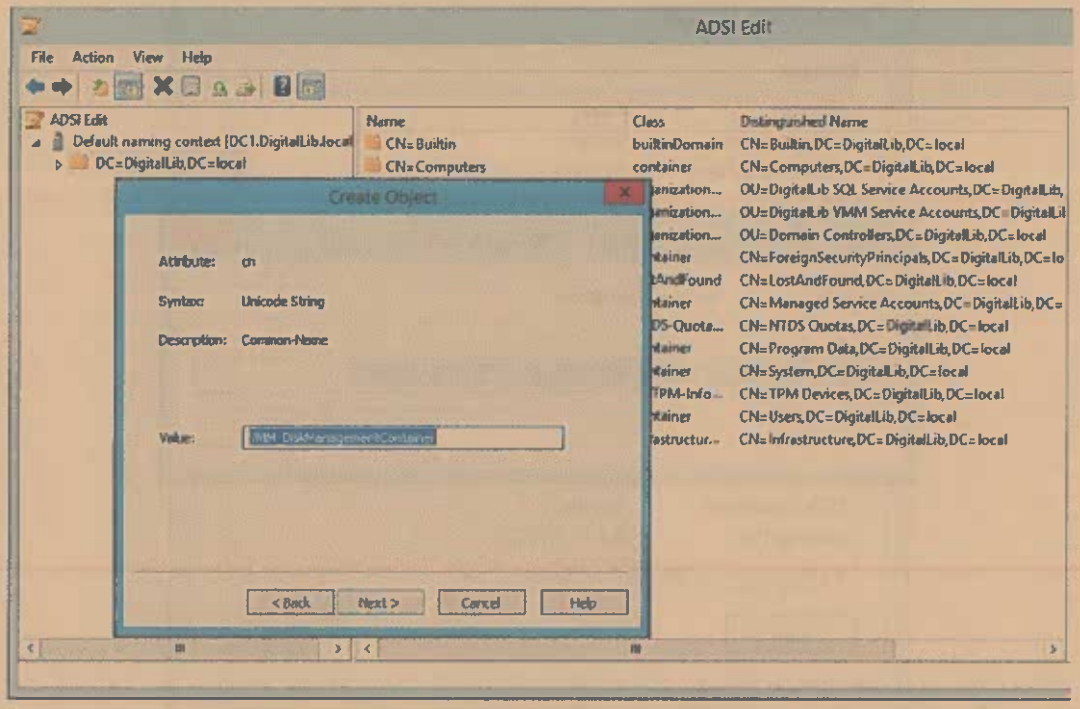

Figura 58 - Dar um nome ao objeto

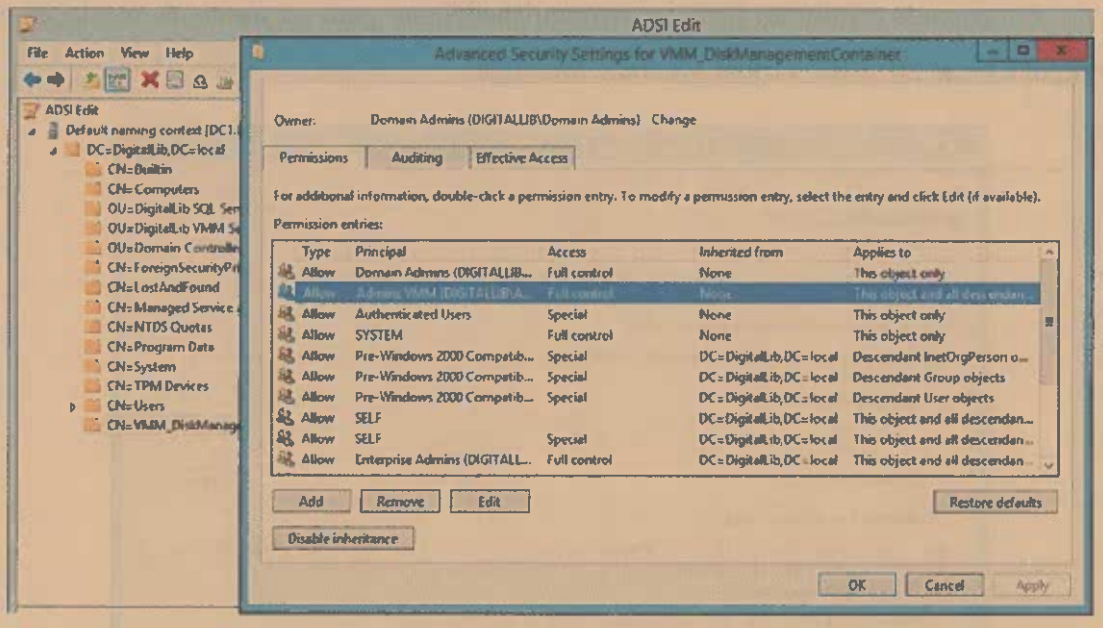

Figura 59 - Dar permissões ao objeto

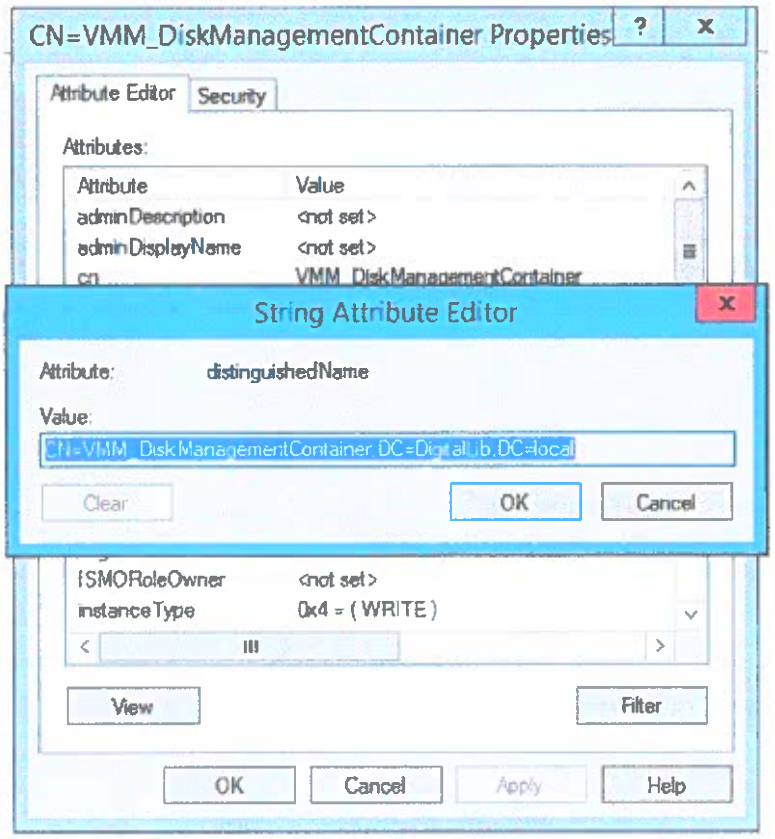

Figura 60 - Campo 'distinguishedName'

Foi copiado o valor do atributo 'distinguishedName' para ser usado na instalação do System Center 2012 R2 Virtual Machine Manager.

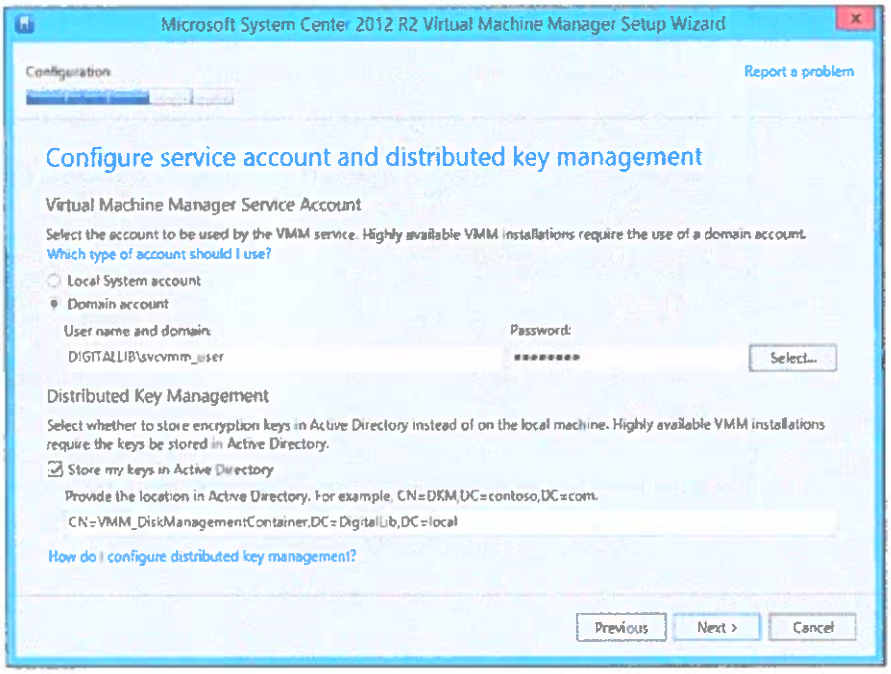

Figura 61 - Configuração da conta de serviço e das 'keys'

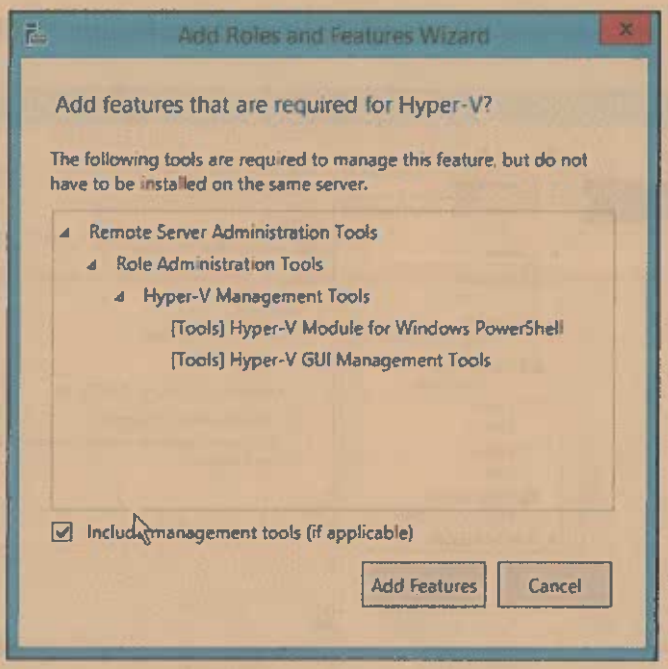

Figura 62 - Instalação da role 'Hyper-V' e das respetivas ferramentas administrativas

Na instalação da role 'Hyper-V' foram selecionadas as placas de rede 'MGMTNET' e 'SQLNET'.

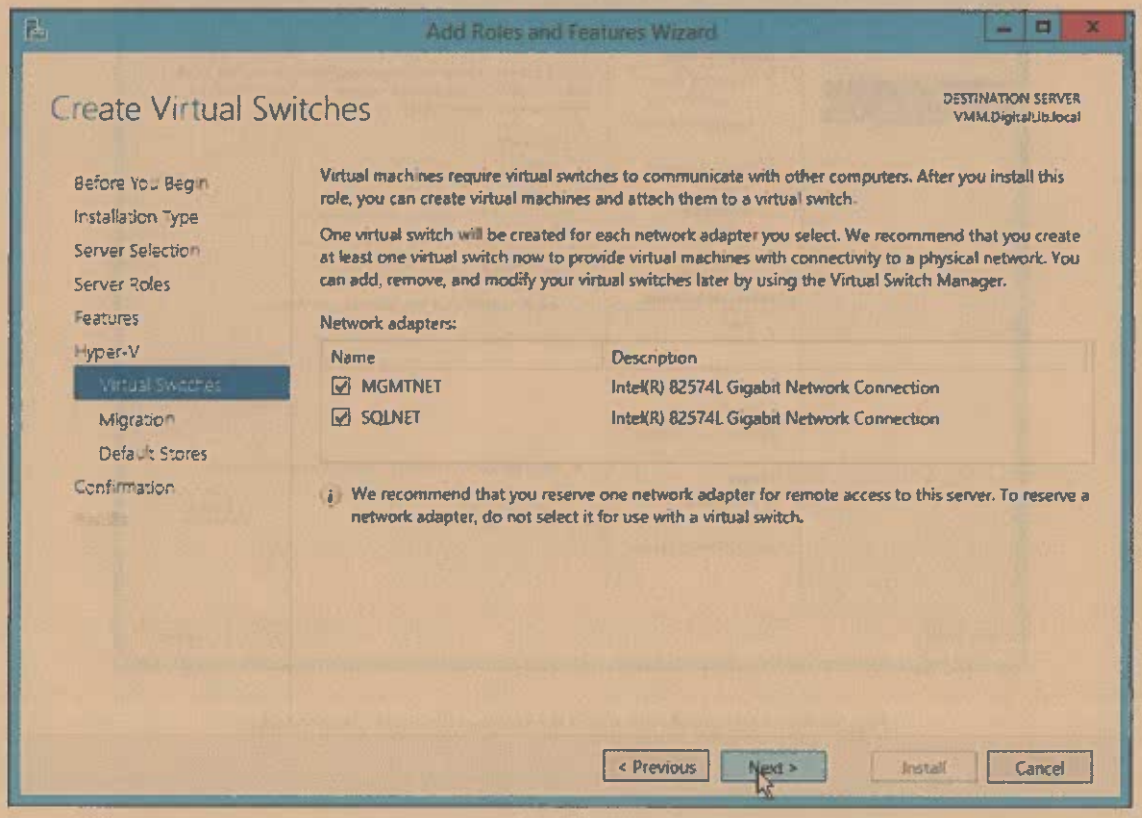

Figura 63 - Criação das Virtual Switches

Após a instalação da role 'Hyper-V', foi instalada a feature 'Multipath I/O'.

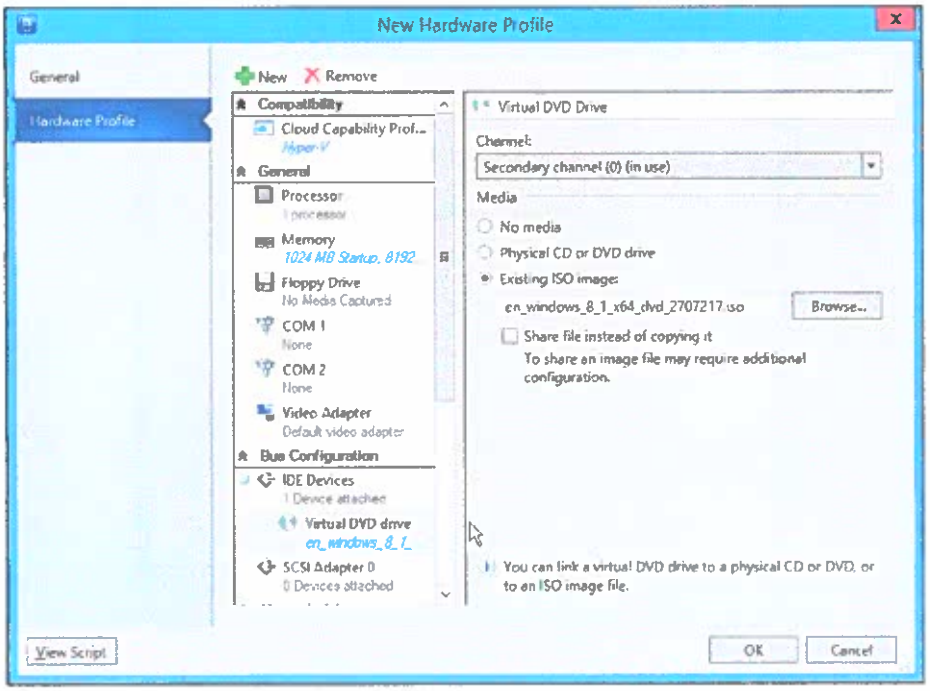

Figura 64 - Criação de um perfil de Hardware

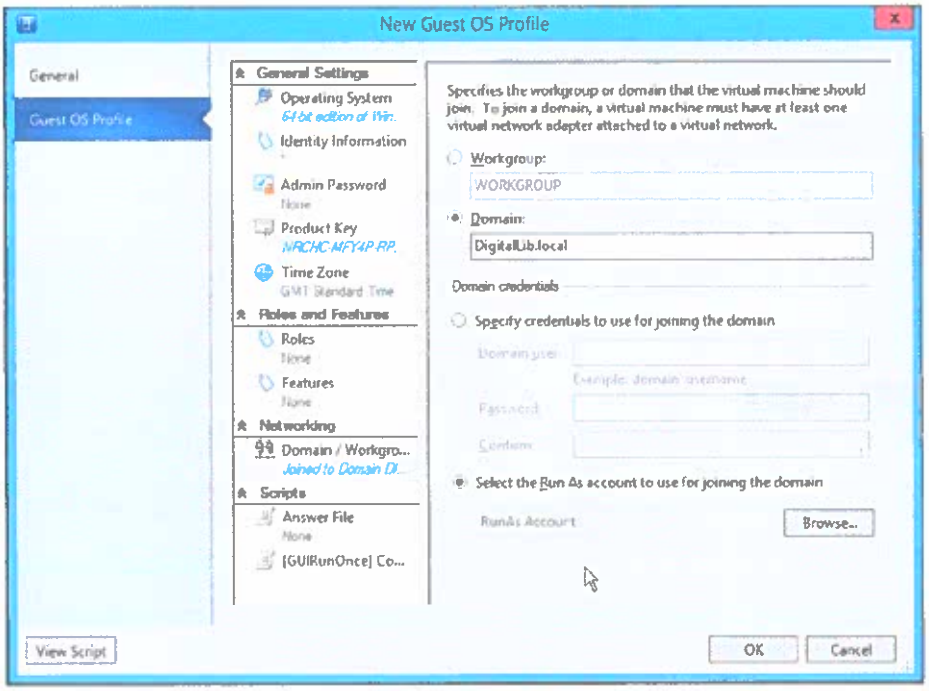

Figura 65 - Criação de um perfil de Sistema Operativo convidado

Para não dar erro na criação de uma máquina virtual, foram dadas permissões ao grupo de administradores 'Admins VMM'.

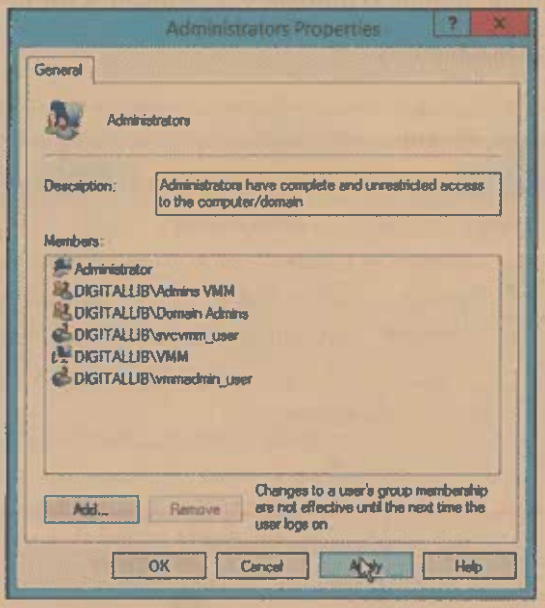

Figura 66 - Adicionar previlégios de administração ao grupo 'Admins VMM'

Depois foi criada uma máquina virtual com base no perfil de *hardware* 'HW_Win8.1'.

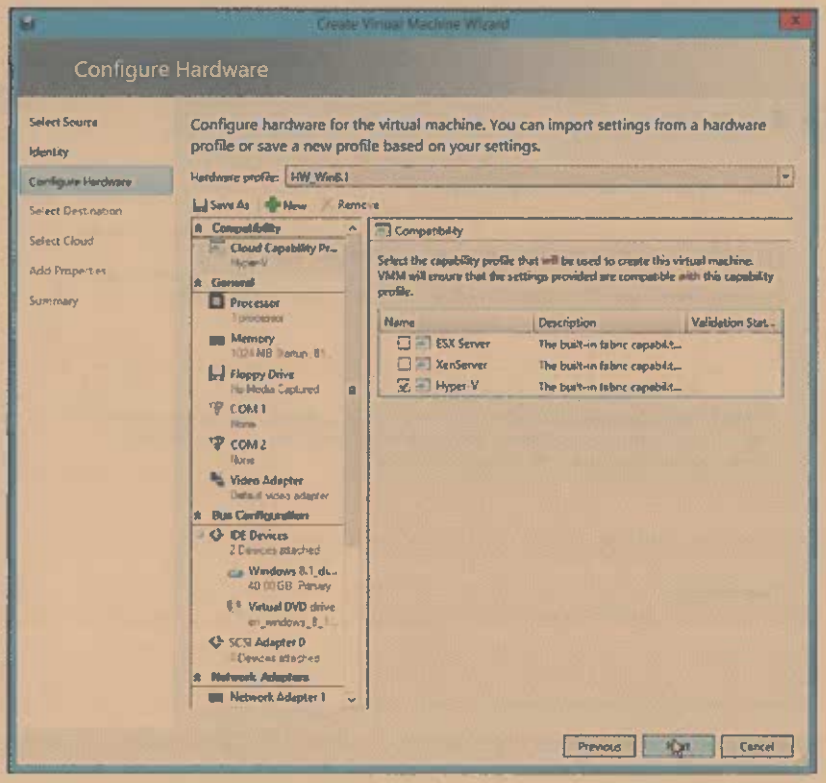

Figura 67 - Criação de uma máquina cliente

f) Criação da máquina cliente

Após criar a máquina cliente Windows 8.1, a mesma foi iniciada e instalada com a versão Professional (figura 66 e 67).

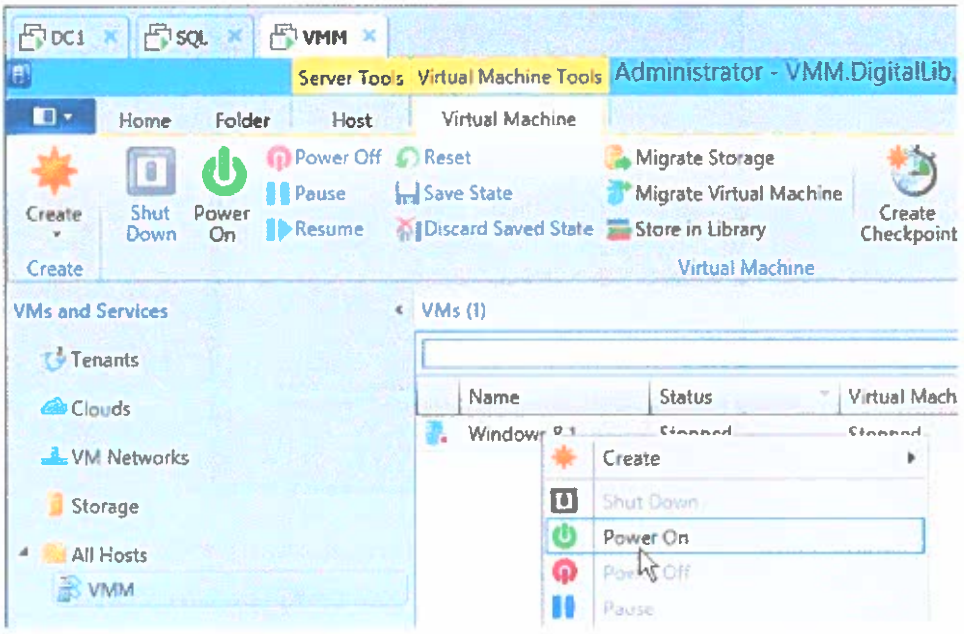

Figura 68 - Inicialização da máquina cliente

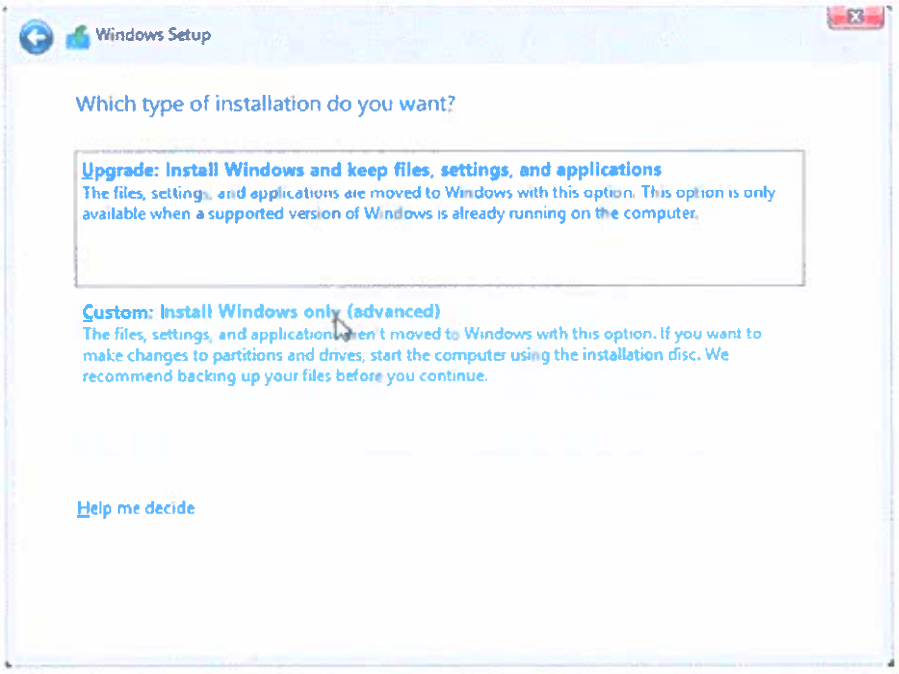

Figura 69 - Instalação do Windows 8.1

Desligou-se e criou-se uma máquina *template* a partir da máquina virtual 'Windows 8.1'.

Figura 70 - Criação de um template

Com o *template* configurado, foi então criada uma máquina cliente onde irá ficar alojada a aplicação da biblioteca digital.

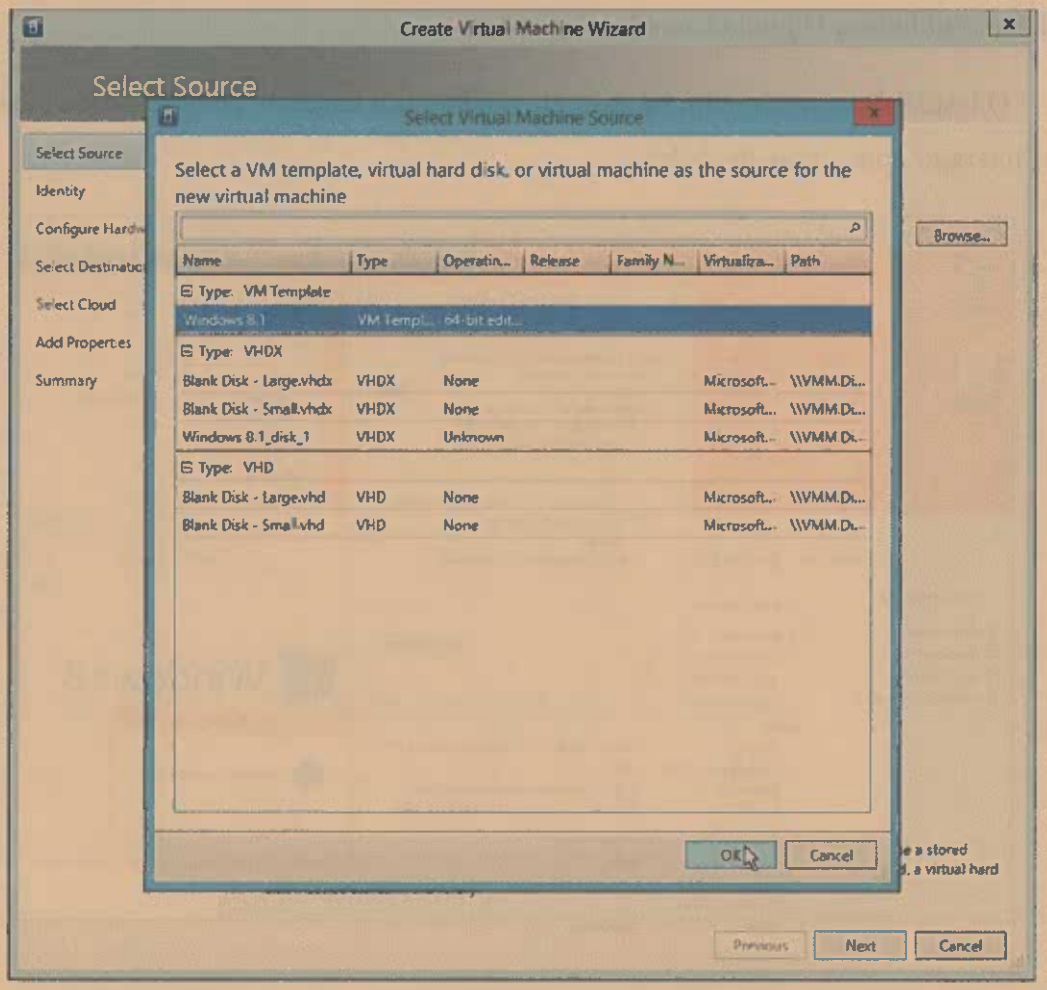

Figura 71 - Criação da máquina cliente com base no template 'Windows 8.1'

Instalação e acesso à aplicação SkyDrive na máquina cliente (figura 70).

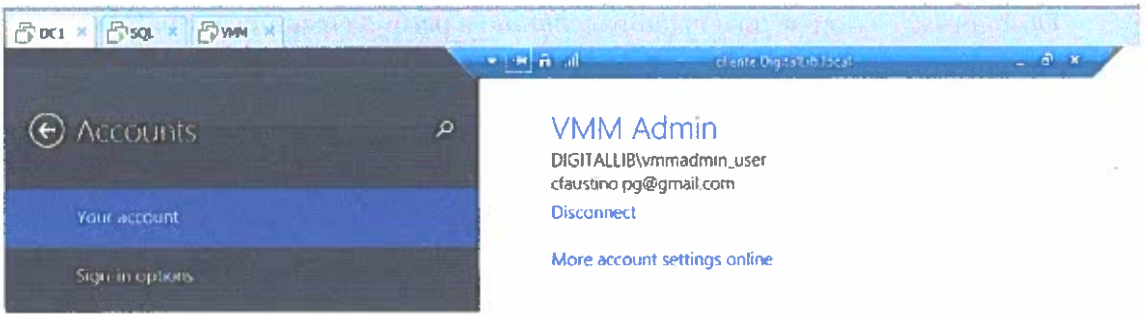

Figura 72 - Conta Microsoft para acesso à SkyDrive

A máquina cliente foi ligada ao domínio 'DigitalLib.local' e, na figura 71, foi testada a ligação ao servidor de base de dados através de um ficheiro do tipo 'Data Link'. Foi colocado o nome do servidor, com acesso por autenticação integrada e com a base de dados da Biblioteca Digital chamada 'DigitalLib'.

O teste foi concluído com sucesso provando que a máquina cliente tem permissões para interagir com a base de dados.

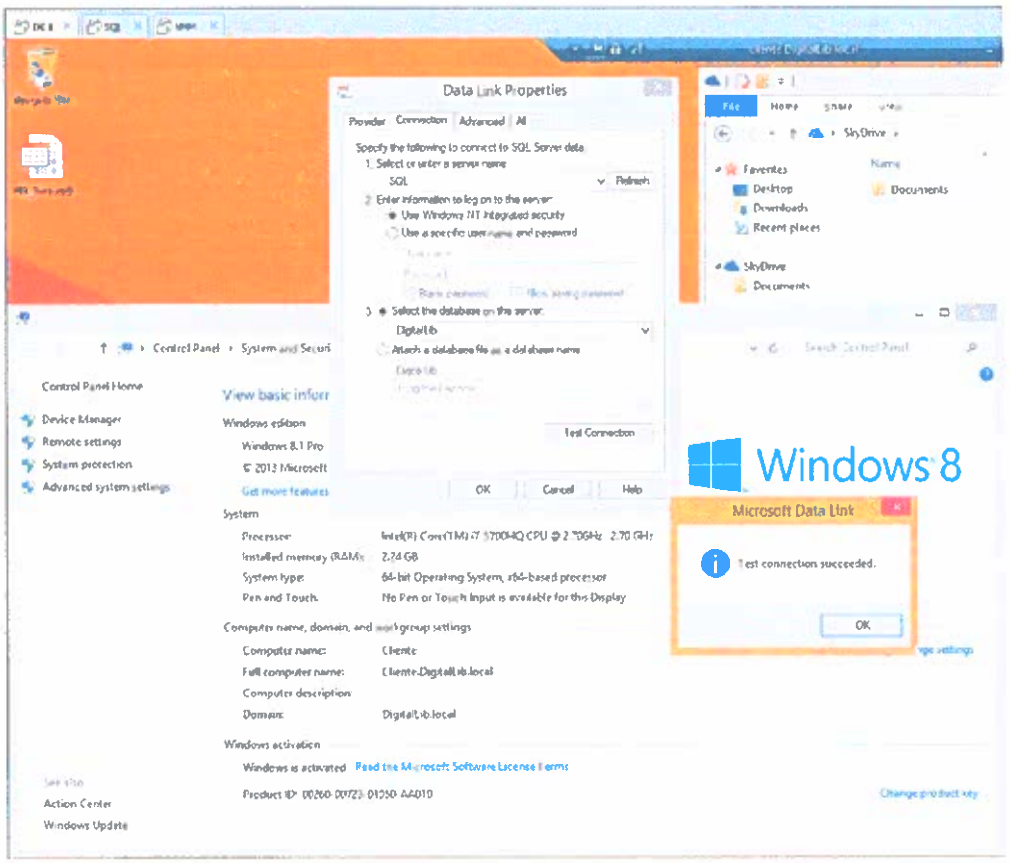

Figura 73 - Máquina cliente e acesso à base de dados

No Active Directory Users and Computers foi criado um utilizador chamado 'celiofaustino' (Célio Faustino) para acesso à máquina cliente, ao qual não foi adicionado ao grupo de administradores.

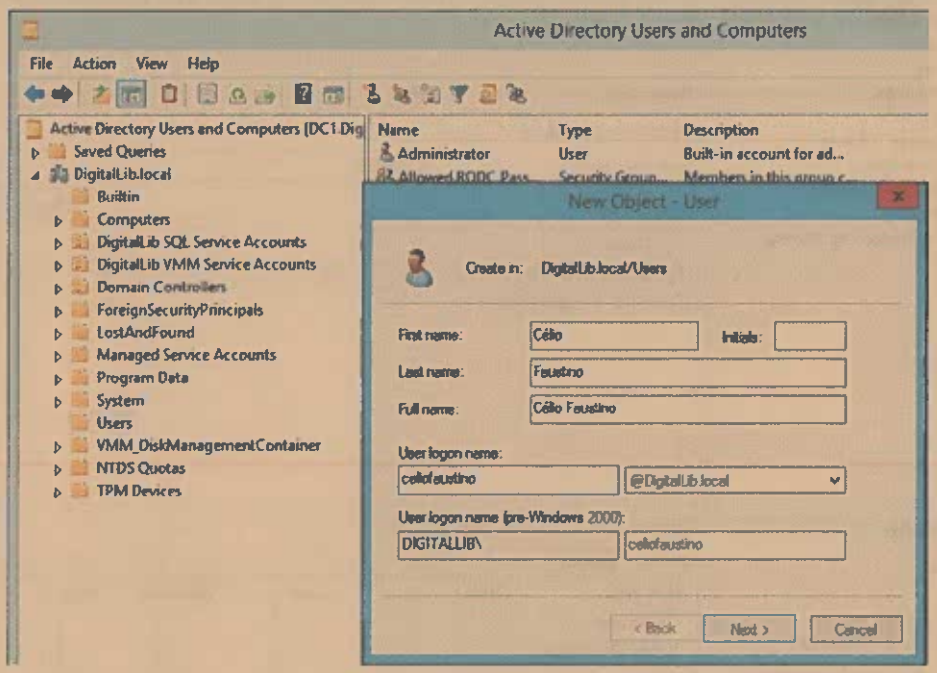

Figura 74 - Criação de utilizador

Criação do 'login' de acesso à base de dados do utilizador 'celiofaustino' e respetiva configuração.

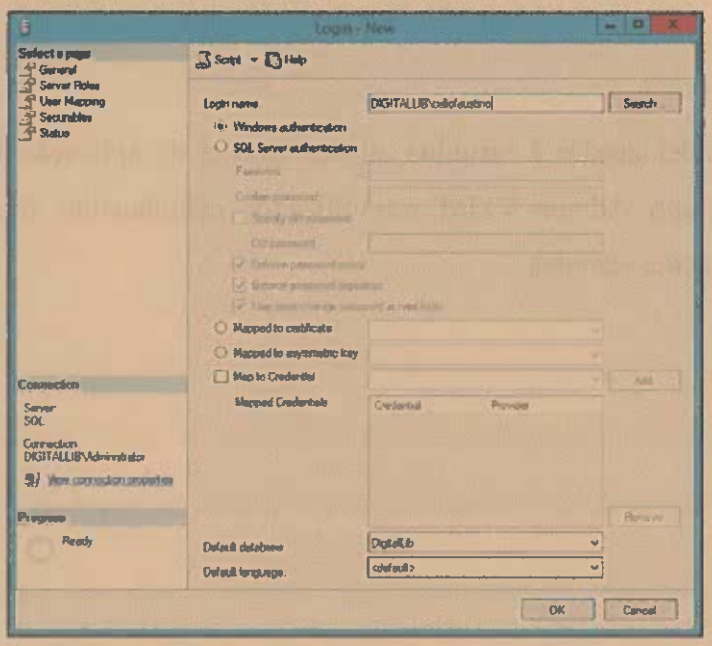

Figura 75 - Configuração do utilizador

Foram dadas permissões de execução de operações na base de dados DigitalLib para o grupo 'Admins VMM' e 'celiofaustino'.

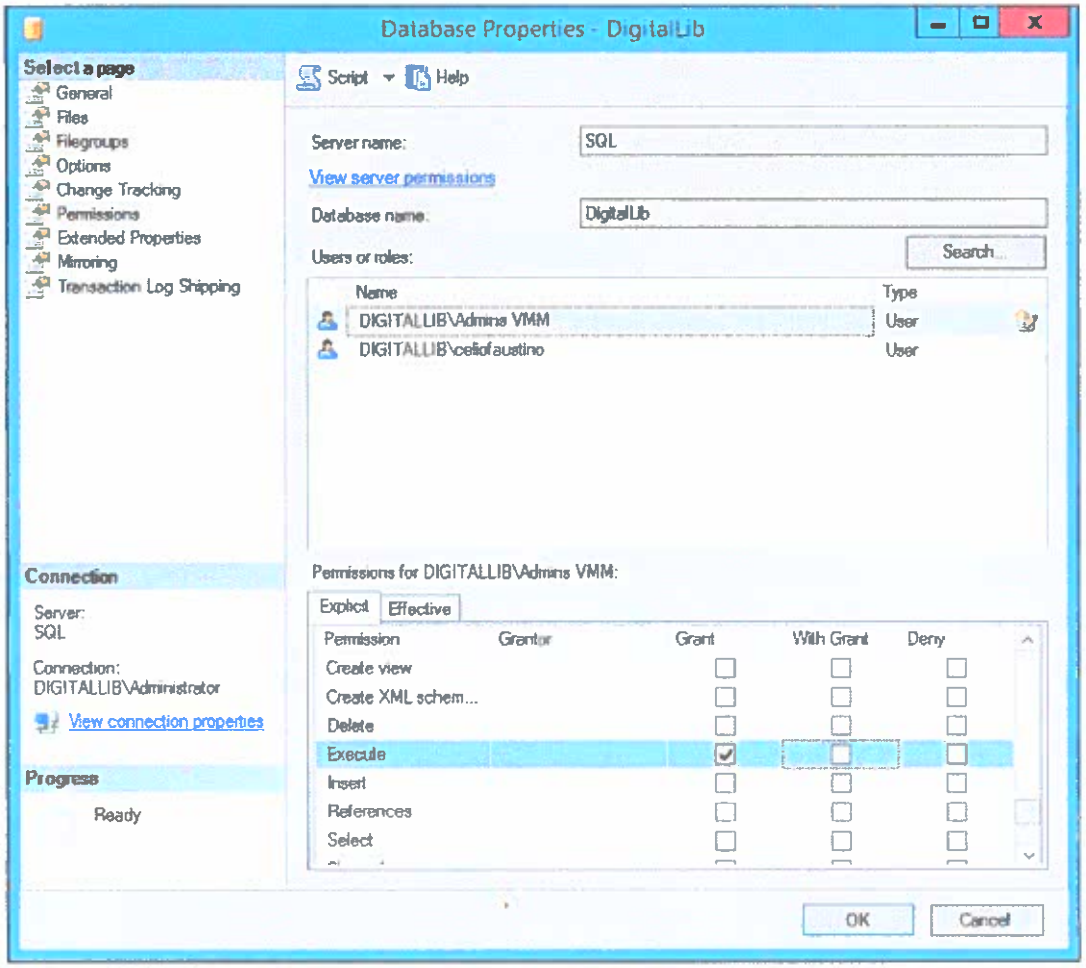

Figura 76 - Permissões de execução

Para se poder aceder à máquina cliente através da aplicação Remote Desktop Connection, o grupo 'Admins VMM' e o utilizador 'celiofaustino' foram configurados para permitir ligações remotas.

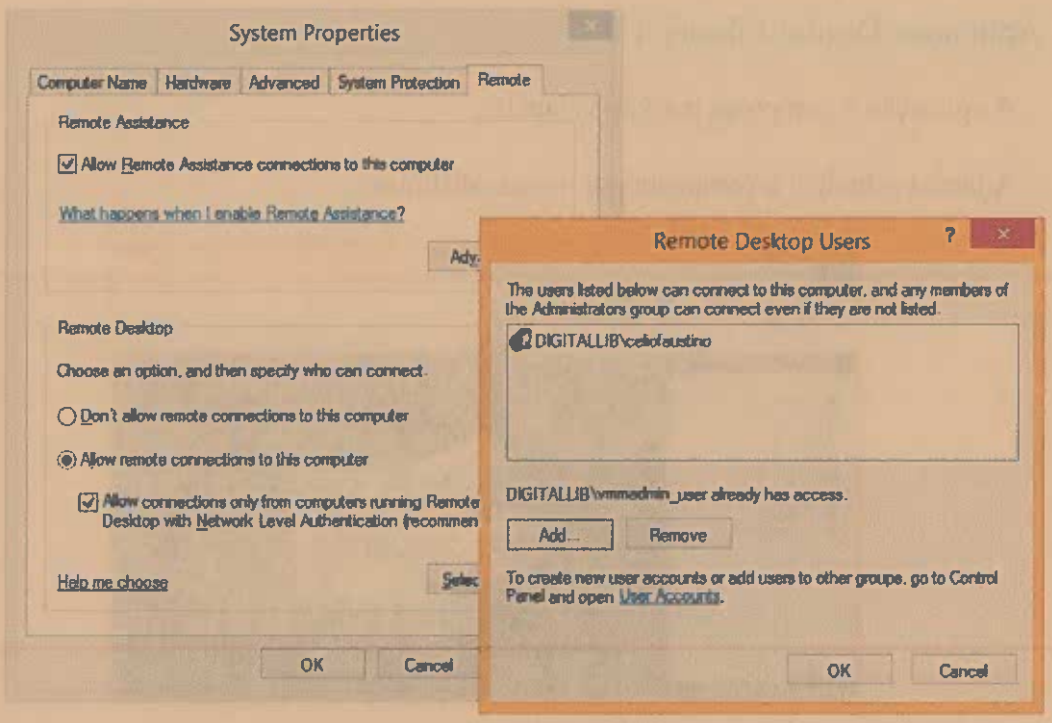

Figura 77 - Configuração de permissão de acesso remoto

Às placas de rede dos servidores SQL e VMM foram efetuadas as seguintes alterações:

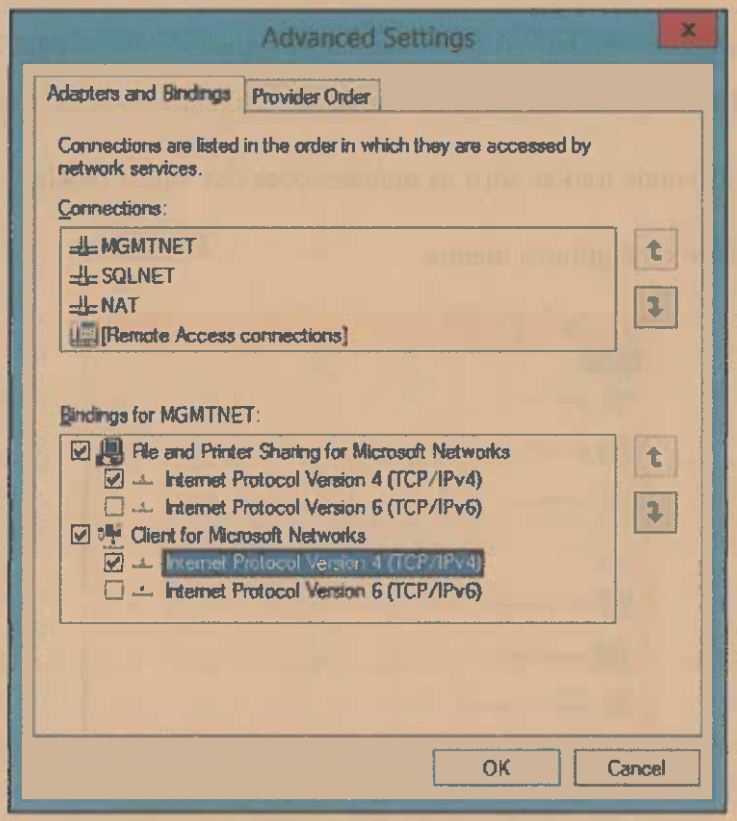

Figura 78 - Ordem das placas de rede

3. Aplicação Digital Library 1.0

A aplicação é composta por várias janelas.

A janela principal é composta por 4 áreas distintas:

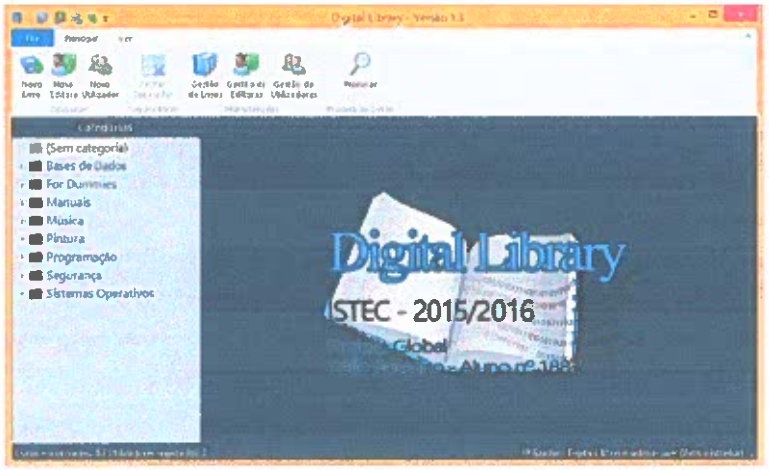

Figura 79 - Visão geral da aplicação

- Uma *Ribbon* (faixa) com o menu *File*, botões de manutenção, pesquisa e outros.

- Uma *Treeview* (à esquerda) onde são expostas as categorias e subcategorias inseridas.

- Uma *StatusBar* (em baixo) onde são dadas algumas informações, nomeadamente qual é o utilizador que está com sessão iniciada no momento e se é ou não administrador.

- E no centro onde irão se abrir as manutenções das várias tabelas.

No menu *File* estão os seguintes menus:

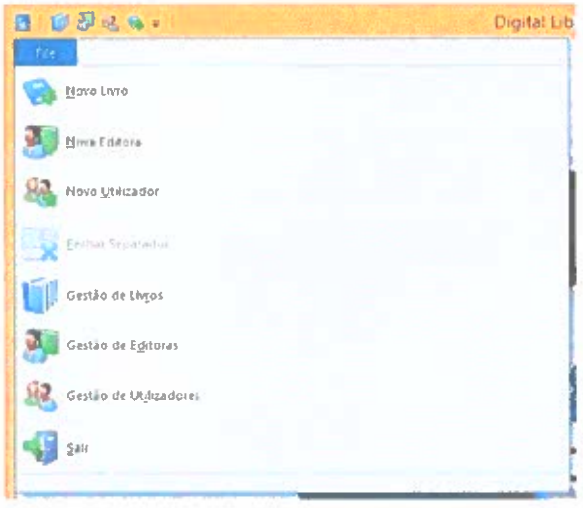

Figura 80 - Menu 'File' da aplicação

Menu Novo Livro – Cria um novo registo na tabela de livros. Ver anexo frmMain.cs no bloco 203-255.

Menu Nova Editora – Cria um novo registo na tabela de editoras. Ver anexo frmMain.cs no bloco 621-648.

Menu Novo Utilizador – Cria um novo registo na tabela de utilizadores. Ver anexo frmMain.cs no bloco 723-751.

Menu Fechar Separador – Fecha um separador aberto. Ver anexo frmMain.cs no bloco 480-493.

Menu Gestão de Livros – Abre um separador com a lista de livros mostrados pela categoria selecionada à esquerda. Ver anexo frmMain.cs no bloco 1328-1391.

Menu Gestão de Editoras – Abre um separador com a lista de editoras. Ver anexo frmMain.cs no bloco 1463-1498.

Menu Gestão de Utilizadores – Abre um separador com a lista de utilizadores com acesso a esta aplicação. Ver anexo frmMain.cs no bloco 1505-1540.

Menu Sair – Sai da aplicação. Ver anexo frmMain.cs no bloco 951-958.

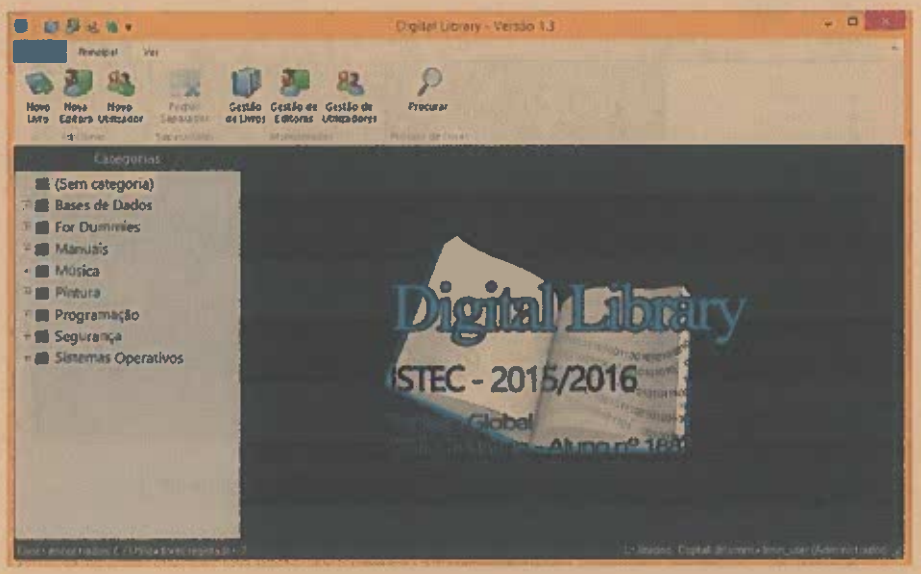

Figura 81 - Ecrã principal em modo de administrador – Separador 'Principal'

Botão Novo Livro – Cria um novo registo na tabela de livros. Ver anexo frmMain.cs no bloco 203-255.

Botão Nova Editora – Cria um novo registo na tabela de editoras. Ver anexo frmMain.cs no bloco 621-648.

Botão Novo Utilizador – Cria um novo registo na tabela de utilizadores. Ver anexo frmMain.cs no bloco 723-751.

Botão Fechar Separador – Fecha um separador aberto. Ver anexo frmMain.cs no bloco 480-493.

Botão Gestão de Livros – Abre um separador com a lista de livros mostrados pela categoria selecionada à esquerda. Ver anexo frmMain.cs no bloco 1328-1391.

Botão Gestão de Editoras – Abre um separador com a lista de editoras. Ver anexo frmMain.cs no bloco 1463-1498.

Botão Gestão de Utilizadores – Abre um separador com a lista de utilizadores com acesso a esta aplicação. Ver anexo frmMain.cs no bloco 1505-1540.

Botão Procurar – Abre um separador com a lista de livros pesquisados não tendo em conta a categoria selecionada na lista à esquerda. Ver anexo frmMain.cs no bloco 1398-1456.

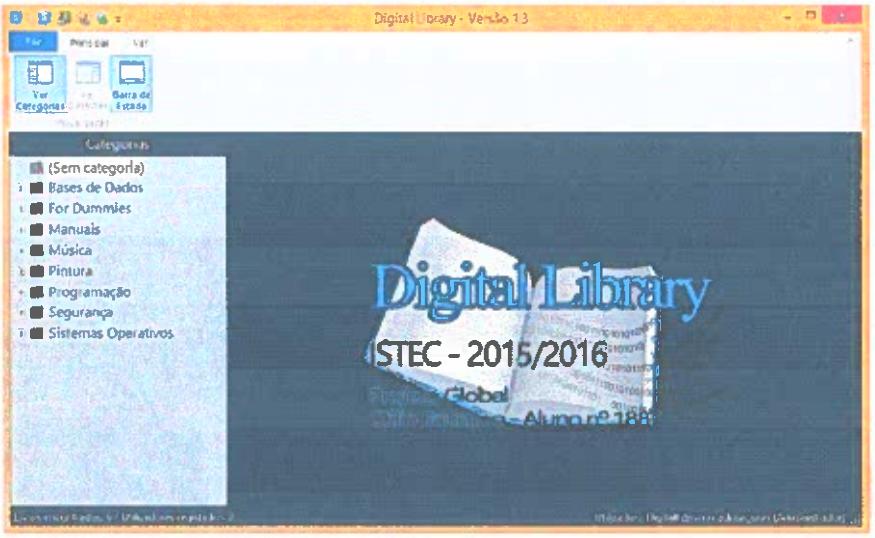

Figura 82 - Ecrã principal em modo de administrador – Separador 'Ver'

Botão Ver Categorias – Mostra ou esconde o controlo TreeView onde são mostradas as categorias (à esquerda). Ver anexo frmMain.cs no bloco 528-532.

Botão Ver Detalhes – Mostra ou esconde o painel de detalhes quando um separador de livros ou uma pesquisa estiver aberta. Ver anexo frmMain.cs no bloco 533-548.

Botão Barra de Estado – Mostra ou esconde a barra de estado (em baixo). Ver anexo frmMain.cs no bloco 616-620.

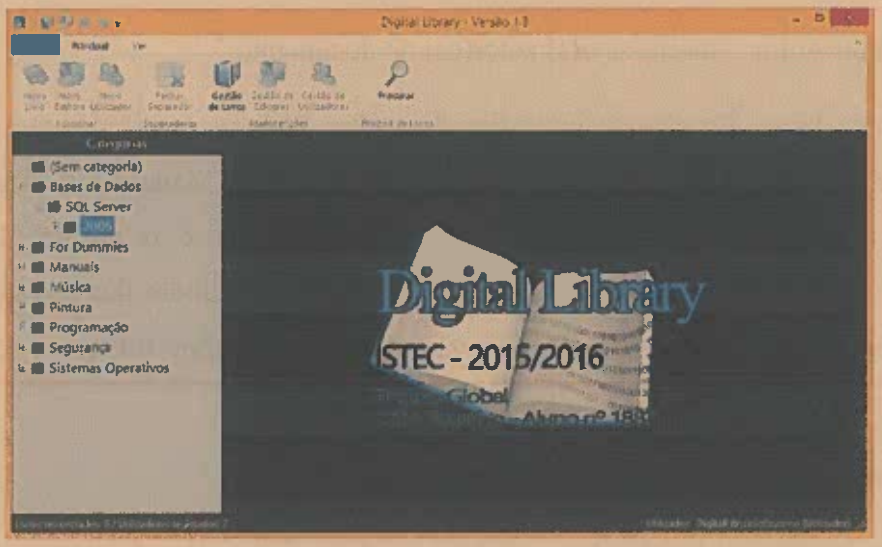

Figura 83 - Ecrã principal em modo de utilizador

Em modo de utilizador normal, o utilizador apenas tem acesso a visualizar os registos de livros e/ou descarregar os respetivos livros (documentos PDF e outros ficheiros). Os únicos botões a que o utilizador tem acesso são: o botão 'Gestão de Livros' e o botão 'Procurar'. No separador 'Ver' o acesso é igual ao utilizador com acesso de administração.

Ao clicar no botão aparece a seguinte janela (ver anexo frmMain.cs no bloco 203-255.):

Figura 84 - Janela de criação de novo livro

Nesta janela são inseridos os valores de identificação de um novo livro ou outro tipo de documento conforme o que é escolhido no campo Tipo.

Campo Referência – Insere-se a referência que se pretende dar ao registo.

Campo Título – Insere-se o título do documento.

Campo Autor – Insere-se o(s) autor(es) do documento.

Campo Tipo – Escolhe-se qual o tipo de registo. Pode ser escolhido entre os valores existentes: 'Livro', 'Link', 'Jornal', 'Revista', 'Imagem' e 'Manuscrito'. Caso o valor escolhido seja 'Livro', o campo Editora fica disponível e o campo 'Link' fica indisponível. Caso o valor escolhido seja 'Link', o campo Editora fica indisponível e o campo Link fica disponível. Nos restantes valores os campos Editora e Link ficam indisponíveis.

Campo Editora – Lista de editoras registadas.

Campo Link – Ligação de uma página *web*, por exemplo, de onde o documento proveio.

Campo Data – Data em que o documento foi criado ou publicado.

Campo Categoria – Categoria a que pertence o documento.

Campo Resumo – Pequeno resumo do documento.

Botão  - Serve para abrir a janela de criação de uma nova editora. Ver anexo frmManBooks.cs no bloco 136-165.

Botão  - Serve para selecionar uma categoria. Ver anexo frmManBooks.cs no bloco 192-208.

Ao clicar no botão  aparece a seguinte janela (ver anexo frmMain.cs no bloco 621-648):

Figura 85 - Janela de criação de nova editora

Esta janela serve para adicionar uma nova editora.

Campo ID – Valor numérico de identificação do registo numerado de forma automática pela base de dados.

Campo Editor – Nome da editora que se pretende criar.

Botão Adicionar – Cria uma nova editora baseado no valor inserido. Ver anexo frmMain.cs no bloco 621-648.

Botão Cancelar – Cancela a operação.

Ao clicar no botão  aparece a seguinte janela (ver anexo frmMain.cs no bloco 723-751):

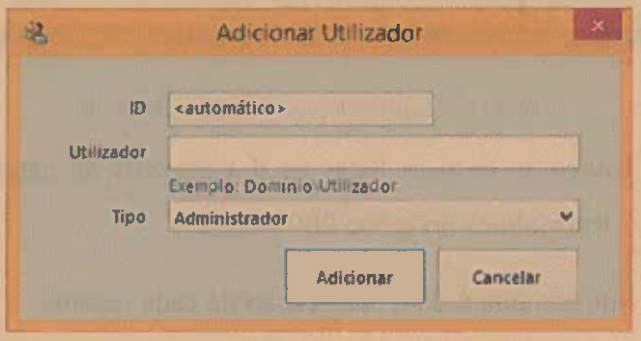

Figura 86 - Janela de criação de novo utilizador

Esta janela serve para adicionar um novo utilizador.

Campo ID – Valor numérico de identificação do registo numerado de forma automática pela base de dados.

Campo Utilizador – Nome do utilizador no domínio. Tem de estar no formato domínio\utilizador.

Campo Tipo – Tipo de utilizador. Se escolher o valor 'Administrador' o utilizador não terá restrições na aplicação. Se escolher o valor 'Utilizador' o utilizador apenas poderá consultar os documentos e transferir o ficheiro respetivo de cada documento.

Botão Adicionar – Cria um novo utilizador baseado nos valores inseridos. Ver anexo frmMain.cs no bloco 723-751.

Botão Cancelar – Cancela a operação.

Ao clicar no botão  aparece o separador Livros (ver anexo frmMain.cs no bloco 1328-1391):

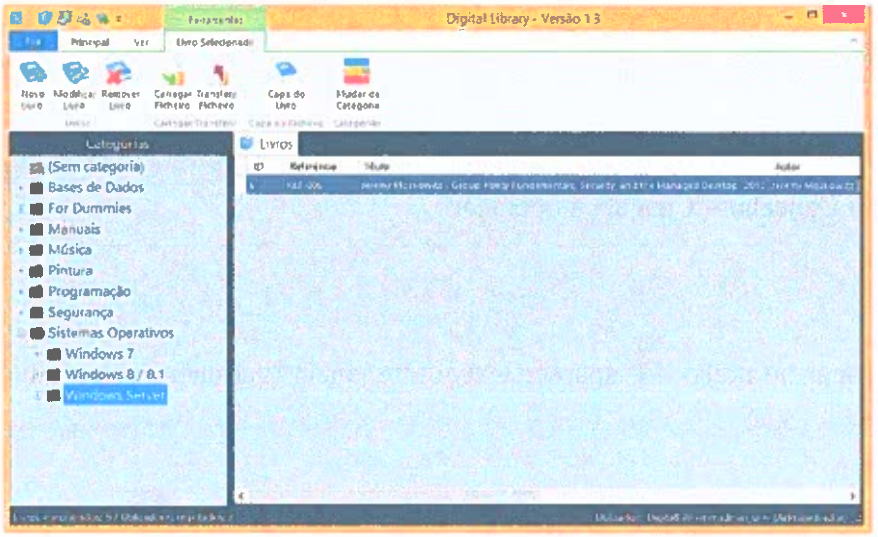

Figura 87 - Separador para a listagem de Livros

O separador Livros serve para listar os documentos da categoria selecionada à esquerda. Ver anexo frmMain.cs no bloco 2003-2013.

Coluna ID – Número único de identificação de cada registo.

Coluna Referência – Referência que se pretende dar ao documento.

Coluna Autor – Autor(es) do documento.

Coluna Editor – Editora que publicou o documento (caso seja um documento do tipo 'Livro').

Coluna Tipo - Pode ser um dos seguintes valores: 'Livro', 'Link', 'Jornal', 'Revista', 'Imagem' ou 'Manuscrito'.

Coluna Link – Caso o tipo de documento seja 'Link', o valor desta coluna será a ligação, por exemplo, para uma página *web*.

Coluna Data – Data em que o documento foi criado ou publicado.

Coluna Categoria – Categoria a que o documento está associado.

Coluna Resumo – Pequeno resumo do documento.

Quando um documento está selecionado, o seguinte separador Ferramentas fica disponível (ver anexo frmMain.cs na linha 1191):

Figura 88 - Separador de Ferramentas para um registo selecionado

Botão Novo Livro – Serve para criar um novo documento. Ver anexo frmMain.cs no bloco 203-255.

Botão Modificar Livro – Serve para modificar um documento existente. Ver anexo frmMain.cs no bloco 256-329.

Botão Remover Livro – Serve para eliminar um documento existente. Ver anexo frmMain.cs no bloco 330-368.

Botão Carregar Ficheiro – Serve para carregar um ficheiro PDF ou de outro tipo para o registo existente na base de dados. Ver anexo frmMain.cs no bloco 369-428.

Botão Transferir Ficheiro – Serve para transferir um ficheiro do registo existente, da base de dados para o computador local. Ver anexo frmMain.cs no bloco 429-471.

Botão Capa do Livro – Serve para atribuir uma ilustração do documento. Ver anexo frmMain.cs no bloco 494- 527.

Botão Mudar de Categoria – Serve para mover um documento de uma categoria para outra. Ver anexo frmMain.cs no bloco 549-615.

Quando dois ou mais documentos estão selecionados, o seguinte separador Ferramentas fica disponível (ver anexo frmMain.cs na linha 1193):

Figura 89 - Separador de Ferramentas para dois ou mais registos selecionados

Botão Novo Livro – Serve para criar um novo documento. Ver anexo frmMain.cs no bloco 203-255.

Botão Modificar Livro – Este botão encontra-se indisponível para quando se tem vários registos selecionados.

Botão Remover Livros – Serve para eliminar os documentos selecionados. Ver anexo frmMain.cs no bloco 330-368.

Botão Carregar Ficheiro – Este botão encontra-se indisponível para quando se tem vários registos selecionados.

Botão Transferir Ficheiros – Serve para transferir todos os ficheiros, dos registos selecionados, da base de dados para uma única pasta do computador local. Ver anexo frmMain.cs no bloco 429-471.

Botão Capa do Livro – Este botão encontra-se indisponível para quando se tem vários registos selecionados.

Botão Mudar de Categoria – Serve para mover todos os documentos selecionados de uma categoria para outra. Ver anexo frmMain.cs no bloco 549-615.

Ao clicar no botão 

aparece o separador Editoras (ver anexo frmMain.cs no bloco 1463-1498):

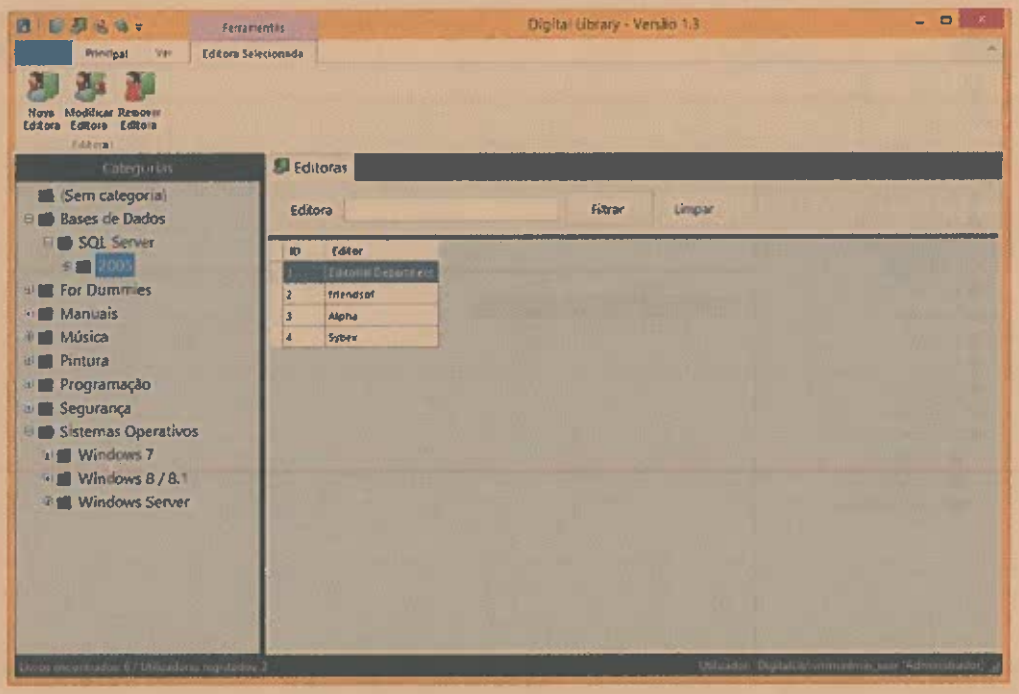

Figura 90 - Separador para a listagem de Editoras

O separador Editoras serve para listar as editoras para usar nos documentos registados.

Coluna ID – Número único de identificação de cada registo.

Coluna Editor – Nome da editora.

Campo Editora – Serve para escrever algo por forma a filtrar os registos na lista.

Botão Filtrar – Serve para filtrar os registos mostrados baseado no texto inserido no campo Editora.

Botão Limpar – Serve para limpar o campo Editora e mostrar tudo novamente.

Ao clicar no botão  aparece o separador Utilizadores (ver anexo frmMain.cs no bloco 1505-1540):

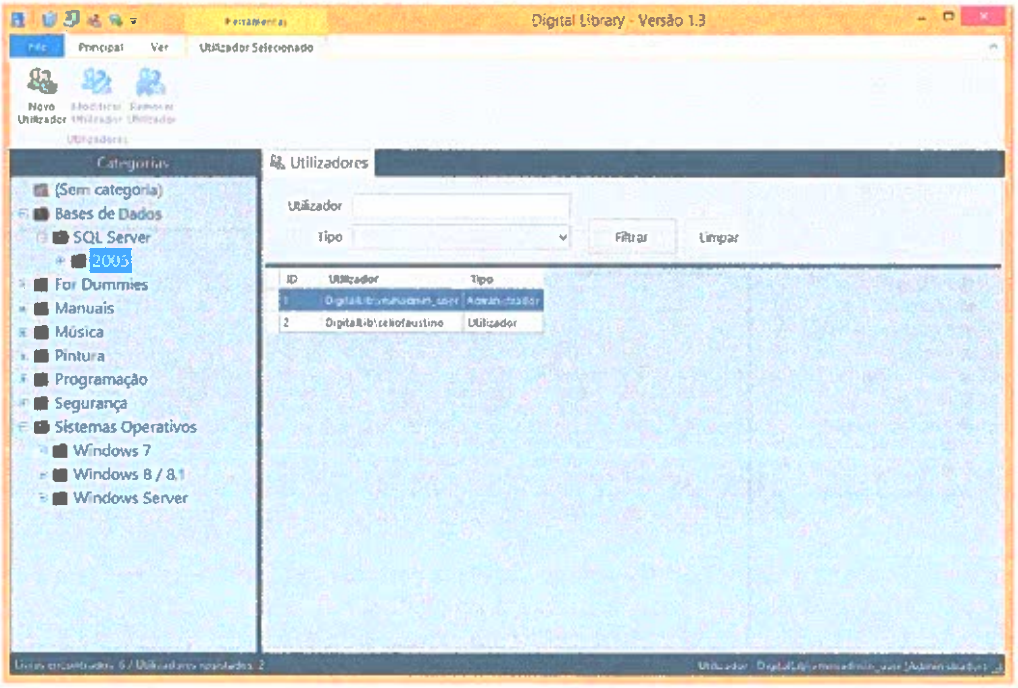

Figura 91 - Separador para a listagem de Utilizadores

O separador Utilizadores serve para listar os utilizadores registados.

Coluna ID – Número único de identificação de cada registo.

Coluna Utilizador – Nome do utilizador no formato Domínio\Utilizador e tem de ser inequívoco.

Coluna Tipo – Tipo de permissão que o utilizador tem na aplicação.

Campo Utilizador – Serve para escrever algo por forma a filtrar os registos na lista.

Campo Tipo – Serve para filtrar por tipos de utilizadores.

Botão Filtrar – Serve para filtrar os registos mostrados baseado no texto inserido nos campos Utilizador e Tipo. Ver anexo ucUsers.cs no bloco 70-74.

Botão Limpar – Serve para limpar os campos Utilizador e Tipo, e mostrar tudo novamente. Ver anexo ucUsers.cs no bloco 108-114.

Ao clicar no botão Procurar aparece o separador Pesquisar (ver anexo frmMain.cs no bloco 1398-1456):

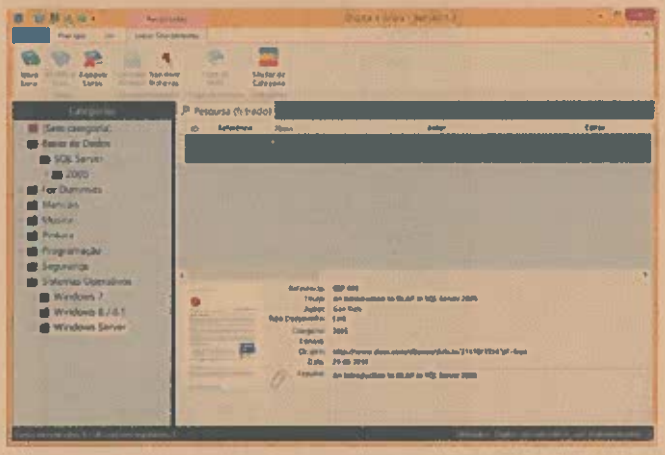

Figura 92 - Separador para a listagem de Livros pesquisados

O separador Pesquisa serve para listar todos os documentos pesquisados não tendo em conta a categoria selecionada à esquerda. Caso não haja registos para o critério de pesquisa, surge o texto 'Não foram encontrados registos. Pesquise de novo.'. Ver anexo ucBooks.cs na linha 162.

Coluna ID – Número único de identificação de cada registo.

Coluna Referência – Referência que se pretende dar ao documento.

Coluna Autor – Autor(es) do documento.

Coluna Editor – Editora que publicou o documento (caso seja um documento do tipo 'Livro').

Coluna Tipo - Pode ser um dos seguintes valores: 'Livro', 'Link', 'Jornal', 'Revista', 'Imagem' ou 'Manuscrito'.

Coluna Link – Caso o tipo de documento seja 'Link', o valor desta coluna será a ligação, por exemplo, para uma página *web*.

Coluna Data – Data em que o documento foi criado ou publicado.

Coluna Categoria – Categoria a que o documento está associado.

Coluna Resumo – Pequeno resumo do documento.

Ao aparecer o separador 'Pesquisar', surge logo uma janela de filtragem dos vários campos (ver anexo ucBooks.cs no bloco 344-383):

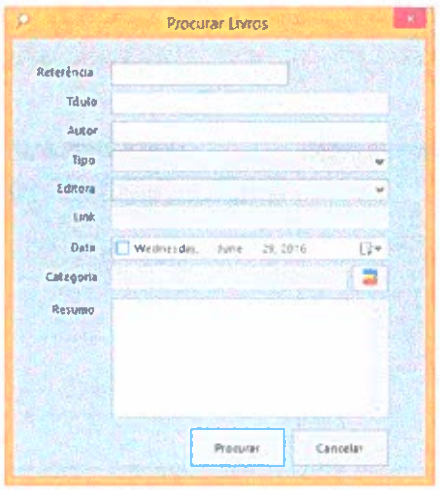

Figura 93 - Janela de filtragem para a pesquisa

Nesta janela são inseridos os valores para a filtragem de livros ou outro tipo de documentos conforme o que é escolhido no campo 'Tipo'.

Campo Referência – Insere-se a referência.

Campo Título – Insere-se o título do documento.

Campo Autor – Insere-se o(s) autor(es) do documento.

Campo Tipo – Escolhe-se qual o tipo de registo. Pode ser escolhido entre os valores existentes: 'Livro', 'Link', 'Jornal', 'Revista', 'Imagem' e 'Manuscrito'. Caso o valor escolhido seja 'Livro', o campo Editora fica disponível e o campo 'Link' fica indisponível. Caso o valor escolhido seja 'Link', o campo Editora fica indisponível e o campo Link fica disponível. Nos restantes valores os campos Editora e Link ficam indisponíveis.

Campo Editora – Lista de editoras registadas.

Campo Link – Ligação de uma página *web*, por exemplo, de onde o documento proveio.

Campo Data – Data em que o documento foi criado ou publicado.

Campo Categoria – Categoria a que pertence o documento.

Campo Resumo – Pequeno resumo do documento.

Botão - Serve para selecionar uma categoria. Ver anexo frmManBooks.cs no bloco 192-208.

Ao clicar no botão aparece a seguinte janela (ver anexo frmMain.cs no bloco 256-329):

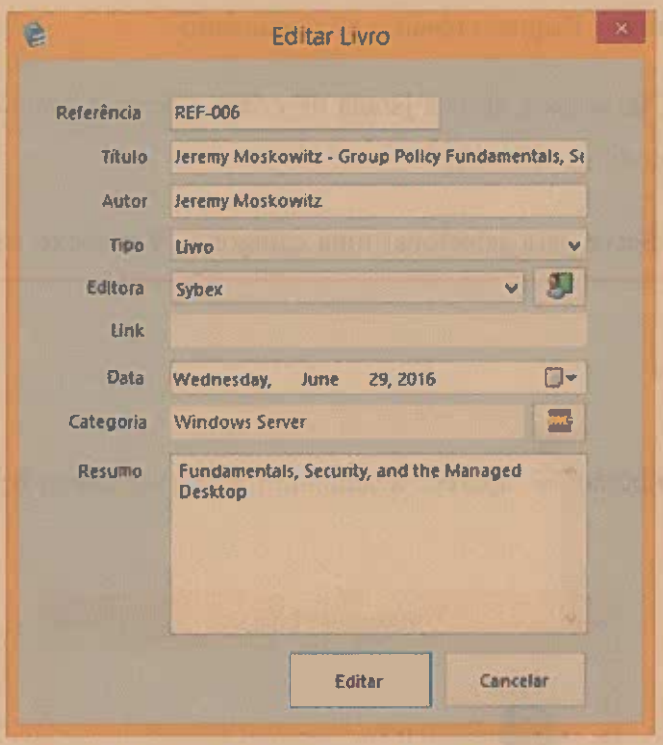

Figura 94 - Janela de modificação de um livro selecionado

Nesta janela são inseridos os valores de identificação do livro selecionado ou outro tipo de documento conforme o que é escolhido no campo 'Tipo'.

Campo Referência – Modifica-se a referência que se pretende dar ao registo.

Campo Título – Modifica -se o título do documento.

Campo Autor – Modifica -se o(s) autor(es) do documento.

Campo Tipo – Escolhe-se qual o tipo de registo. Pode ser escolhido entre os valores existentes: 'Livro', 'Link', 'Jornal', 'Revista', 'Imagem' e 'Manuscrito'. Caso o valor escolhido seja 'Livro', o campo Editora fica disponível e o campo 'Link' fica indisponível. Caso o valor escolhido seja 'Link', o campo Editora fica indisponível e o campo Link fica disponível. Nos restantes valores os campos Editora e Link ficam indisponíveis.

Campo Editora – Lista de editoras registadas.

Campo Link – Ligação de uma página *web*, por exemplo, de onde o documento proveio.

Campo Data – Data em que o documento foi criado ou publicado.

Campo Categoria – Categoria a que pertence o documento.

Campo Resumo – Pequeno resumo do documento.

Botão  - Serve para abrir a janela de criação de uma nova editora. Ver anexo frmManBooks.cs no bloco 136-165.

Botão  - Serve para selecionar uma categoria. Ver anexo frmManBooks.cs no bloco 192-208.

Ao clicar no botão  aparece a seguinte janela (ver anexo frmMain.cs no bloco 330-368):

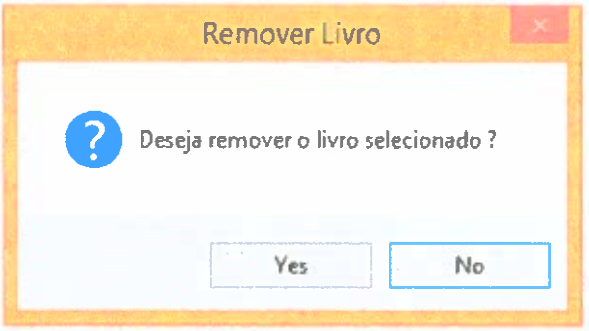

Figura 95 - Janela de confirmação de eliminação do registo

Nesta janela é solicitado ao utilizador para confirmar a eliminação do registo.

Ao clicar no botão  aparece a seguinte janela (ver anexo frmMain.cs no bloco 369-428):

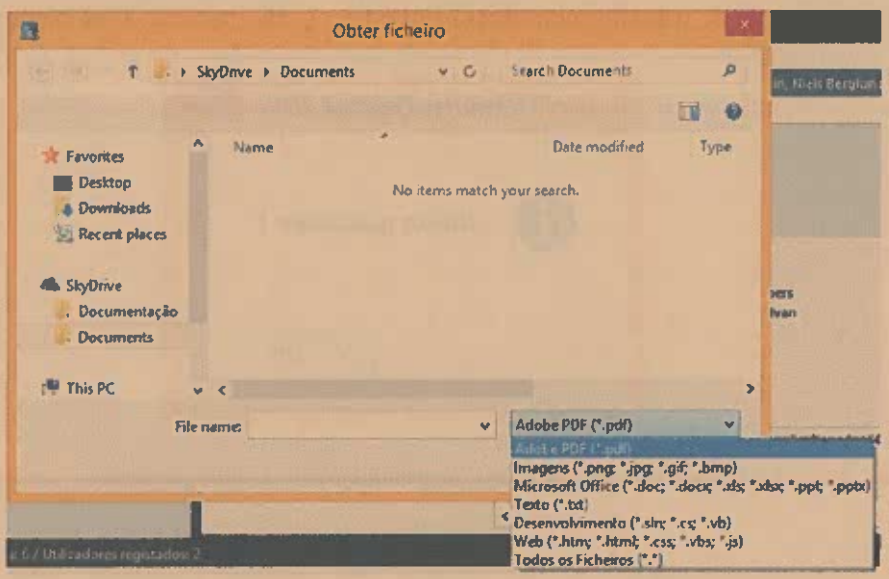

Figura 96 - Caixa de diálogo para obter o ficheiro a carregar

Esta janela permite pesquisar no Windows Explorer pelo documento a carregar para o registo.

Ao clicar no botão  aparece a seguinte janela (ver anexo frmMain.cs no bloco 429-471):

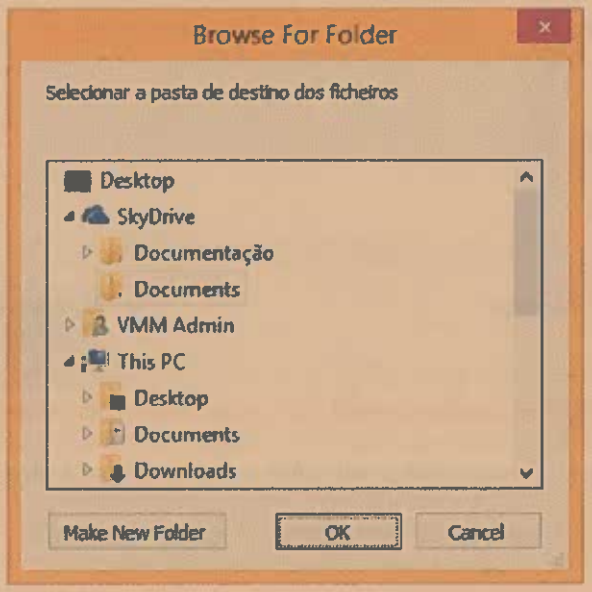

Figura 97 - Janela de pesquisa da diretoria destino do(s) ficheiro(s) a transferir

Após o sucesso de transferência do(s) ficheiro(s), irá aparecer a mensagem:

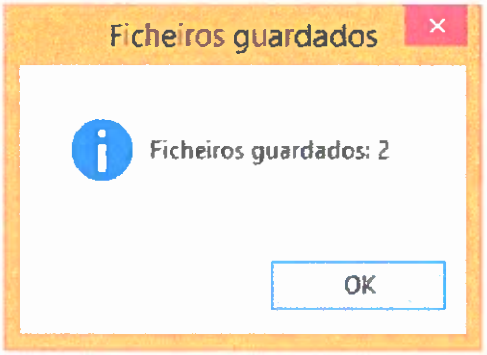

Figura 98 - Mensagem de sucesso de quantos ficheiros foram guardados

Ao clicar no botão  aparece a seguinte janela (ver anexo frmMain.cs no bloco 494-527):

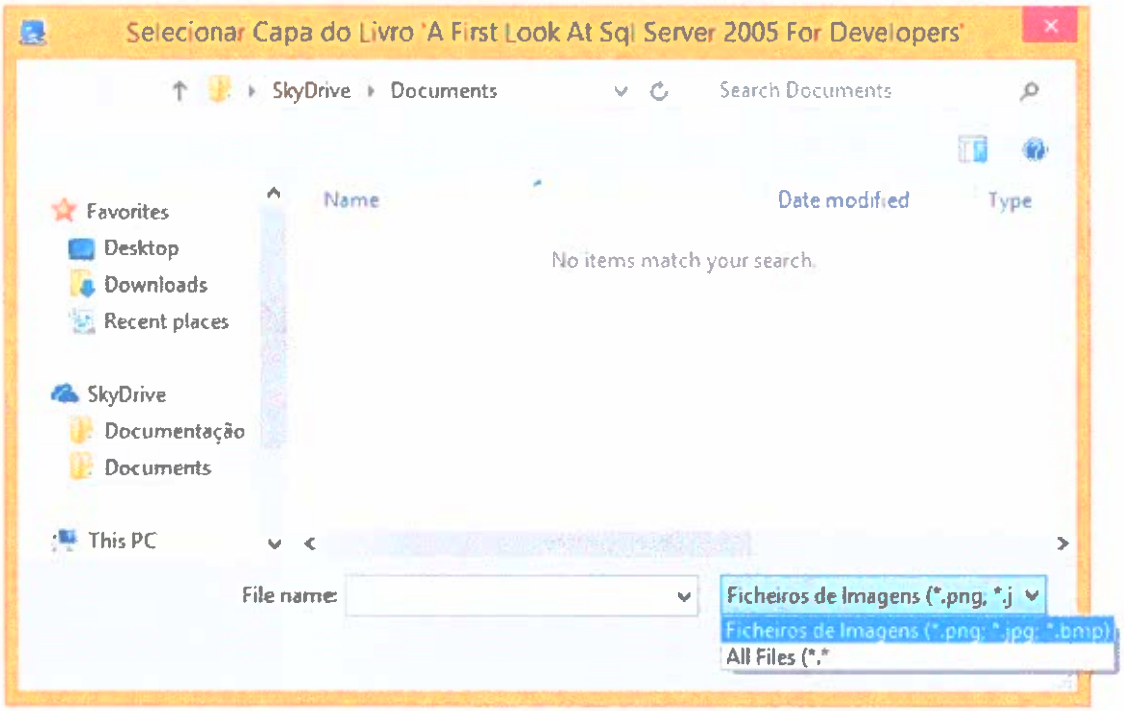

Figura 99 - Caixa de diálogo para obter a imagem da capa do livro ou documento

Ao clicar no botão 

aparece a seguinte janela (ver anexo frmMain.cs no bloco 549-615):

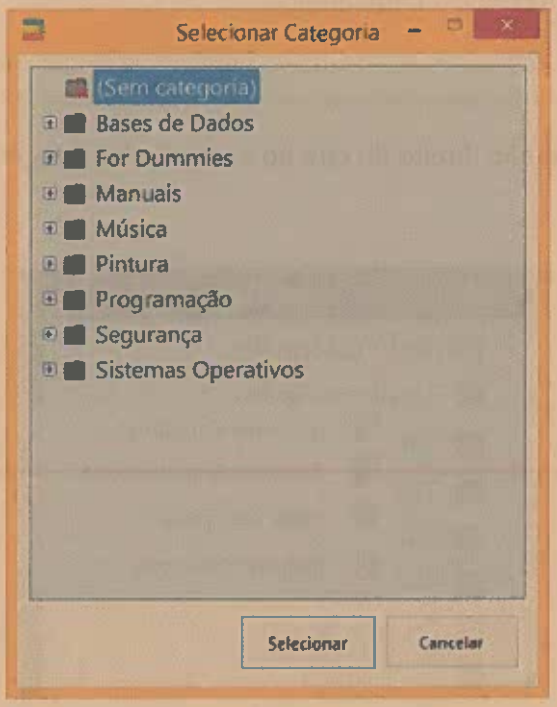

Figura 100 - Janela de seleção de categoria

Esta janela serve para escolher uma outra categoria, por forma a mover um livro ou documento para outra categoria.

No separador Ver, existem três botões para modificar o modo de visualização da aplicação:

Ao clicar no botão 

o painel das categorias fica invisível. Ao clicar de novo no botão 

o painel das categorias fica visível novamente.

Ao clicar no botão 

o painel dos detalhes no separador Livros ou Pesquisa fica invisível. Ao clicar de novo no botão 

o painel dos detalhes no separador Livros ou Pesquisa fica visível novamente.

Ao clicar no botão Barra de Estado a barra de estado fica invisível. Ao clicar de novo no botão Barra de Estado a barra de estado fica visível novamente.

Ao clicar com o botão direito do rato no controlo das categorias, aparece o seguinte menu:

Figura 101 - Menu de contexto com base na seleção da categoria

Menu Adicionar Categoria – Serve para adicionar uma categoria principal (fica na raiz da árvore) (Ver anexo frmMain no bloco 831-857).

Menu Adicionar SubCategoria – Serve para adicionar uma subcategoria à categoria selecionada (Ver anexo frmMain no bloco 858-885).

Menu Editar Categoria – Serve para editar a categoria selecionada (Ver anexo frmMain no bloco 886-914).

Menu Remover Categoria – Serve para eliminar uma subcategoria selecionada. Atenção que quando se remove uma categoria, todas as subcategorias são igualmente removidas e todos os registos de livros ou documentos são recategorizados para (Sem categoria) (Ver anexo frmMain no bloco 915-946).

Ao clicar com o botão direito do rato num registo da listagem dos livros ou das pesquisas, aparece o seguinte menu:

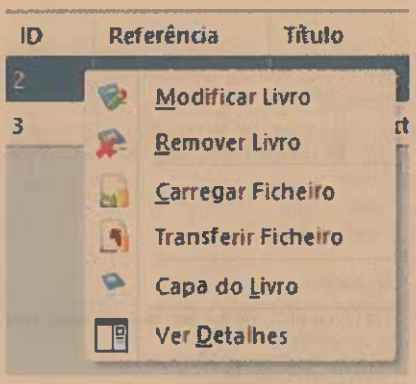

Figura 102 - Menu de contexto com base na seleção de Livros ou documentos

Menu Modificar Livro – Serve para editar um livro ou documento selecionado. Ver anexo frmMain.cs no bloco 256-329.

Menu Remover Livro – Serve para eliminar um livro ou documento selecionado. Ver anexo frmMain.cs no bloco 330-368.

Menu Carregar Ficheiro – Serve para carregar um ficheiro para o livro ou documento selecionado. Ver anexo frmMain.cs no bloco 369-428.

Menu Transferir ficheiro – Serve para transferir um ficheiro para uma diretoria do computador local. Ver anexo frmMain.cs no bloco 429-471.

Menu Capa do Livro – Serve para atribuir uma imagem de capa do livro ou documento. Ver anexo frmMain.cs no bloco 494- 527.

Menu Ver Detalhes – Serve para visualizar ou não o detalhe do registo selecionado. Ver anexo frmMain.cs no bloco 533-548.

Ao clicar no botão 

o painel dos detalhes no separador Livros ou Pesquisa fica visível.

Figura 103 - Painel de detalhes de um registo

O detalhe é composto por todos os campos de um registo, incluindo a imagem de capa (figura 102) e se tem um ficheiro carregado na base de dados (figura 103).

Figura 104 - Imagem de capa do registo

Figura 105 - Imagem representativa de registo com ficheiro carregado

Conclusão

A aplicação Digital Library foi desenvolvida com vista a apresentar uma forma consistente de trabalhar, cooperativamente e em rede com a utilização das tecnologias de informação. Cada livro ou documento, por ser digital, é muito mais fácil de aceder de qualquer lugar e permite, de forma precisa, pesquisar a informação requerida com uma rapidez superior a um livro tradicional.

Para a criação desta aplicação foi adicionada mais informação para uma melhor organização e separação dos vários tipos de documentos a inserir nesta base de dados. A separação por categorias permite ao utilizador encontrar o livro na secção à qual pertence e assim filtrar com mais precisão.

Através da tecnologia de virtualização, pode-se reduzir os elevados custos financeiros que a aquisição de *hardware* necessitaria com uma estrutura destas. As mais recentes versões de software foram utilizadas, por forma a ter uma plataforma estável e para que a aplicação pudesse ser executada em segurança.

Bibliografia

Alemán, P. (2010). *Biblioteca Digital o Virtual.*

Almeida, J. (2011). Virtualização de Servidores. *Computer World.*

Armbrust, M. (2009). *Above the Clouds: A Berkeley View of Cloud Computing.*

Bent, J. W., & Steeg, M. (2012). *The Workbook Company.*

Buyya, R., Vecchiola, C., & Selvi, S. (2013). *Mastering Cloud Computing Foundations and Applications Programming.*

Catteddu, D., & Hogben, G. (2009). *Cloud Computing - Benefits, risks and recomendations for information security.*

Chand, M., & Gold, M. (2002). *A Programmer's Guide to ADO.NET in C#.*

Cooter, M. (2011). *The Ultimate Guide to Cloud Computing.* CloudPro.

Date, C. J. (1986). *Relational Database: Selected Writings.*

Erl, T., Puttini, R., & Mahmood, Z. (2013). *Cloud Computing: Concepts, Technology & Architecture.*

Ferreira, N. (2004). *ADO.NET.*

Galvin, P. (2009). VMWare vSphere vs. Microsoft Hyper-V - A Technical Analysis. *A CTI Strategy WHite Paper.*

Gilster, R. (2014). *CompTIA Cloud+ CV0-001 In Depth.*

Goldberg, R. (1974). Survey of Virtual Machine Research.

Joseph, R. (2007). *"An Introduction to Mainframes".*

Jozwiak, R., & Bloor, R. (2011). *Moving to the Cloud with Pervasive PSQL.*

Kline, K., & Kline, D. (2001). *"SQL in a Nutshell - A Desktop Quick Reference".* O'Reilly.

Ky, j. (2013). *C#: A Beginner's Tutorial.*

Leon, A., & Leon, M. (1999). *SQL - A Complete Reference.*

Malik, N. (2007). *Pro ADO.NET 2.0.*

Mann, A. (2006). Virtualization 101: Technologies, Benefits and Chalenges.

Marinescu, D. C. (2013). *Cloud Computing - Theory and Practice.*

Marshall, D. (2011). Virtualization Report. *InfoWorld.*

Millington, B., & Kauffman, J. (2009). *Beginning ASP.NET 2.0 and Databases.*

Okano, M., & Andrade, F. (2008). O Impacto da Virtualização nas Empresas.

Rao, M. (2015). *Cloud Computing.*

Sosinsky, B. (2011). *Cloud Computing Bible.*

Strachey, C. (1959). Time Sharing in Large Fast Computers.

Troelsen, A. (2012). *"Pro C# 5.0 and the .NET 4.5 Framework"*. APress.

Trukhnov, A. K. (2008). *"SQL Bible"*. John Wiley & Sons © 2003.

Veras, m. (2011). *Virtualização - Componente Central do Datacenter.*

Wilton, P., & Colby, J. (2005). *Beginning SQL.*

Anexos

a) SQL Server – Diagrama da base de dados

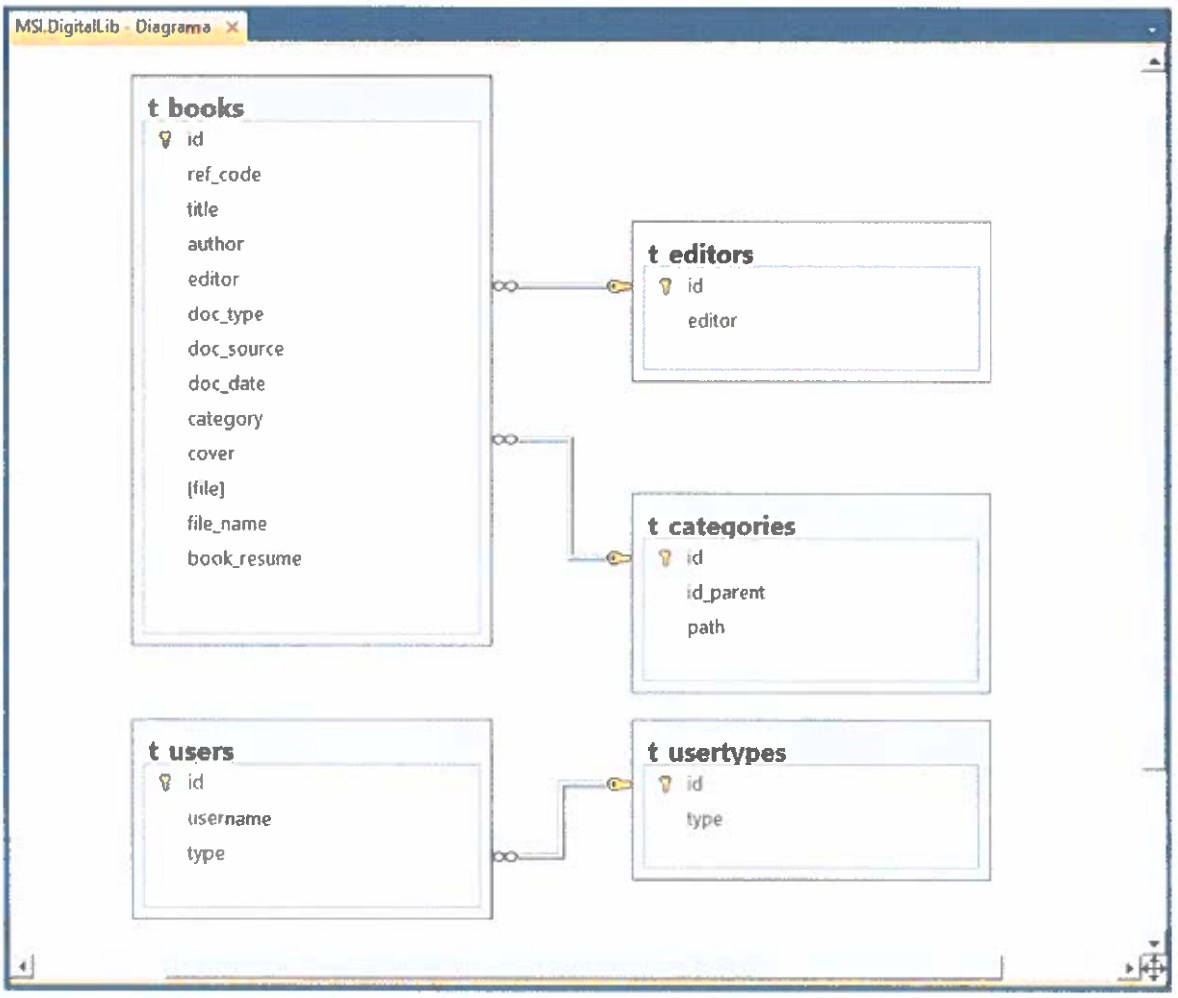

Figura 106 - Diagrama geral da base de dados da aplicação

b) SQL Server – Tabelas

Tabela de Livros

t books

Column Name	Data Type	Allow Nulls
id	int	☐
ref_code	varchar(50)	☐
title	varchar(200)	☐
author	varchar(50)	☐
editor	int	☑
doc_type	int	☑
doc_source	varchar(500)	☑
doc_date	datetime	☑
category	int	☑
cover	image	☑
[file]	image	☑
file_name	varchar(100)	☑
book_resume	varchar(2000)	☐
		☐

Figura 107 - Tabela de Livros

Campo	*Tipo de dados*	*Descrição*
id	int	Identificação inequívoca do registo *Este campo é auto incremental do tipo número inteiro.*
ref_code	varchar(50)	Referência do livro *Este campo é do tipo texto com um máximo de 50 carateres.* *Não suporta valores nulos.*
title	varchar(200)	Título do livro *Este campo é do tipo texto com um máximo de 200 carateres.* *Não suporta valores nulos.*
author	varchar(50)	Autor(es) do livro *Este campo é do tipo texto com um máximo de 50 carateres.* *Não suporta valores nulos.*
editor	int	Editora do livro (se doc_type = 1 'Livro') *Este campo é do tipo número inteiro.* *Chave estrangeira com a tabela t_publishers com o campo id.*
doc_type	int	Tipo de documento *Este campo é do tipo número inteiro.*
doc_source	varchar(500)	Ligação Web (se doc_type = 2 'Link')

		Este campo é do tipo texto com um máximo de 500 carateres.
doc_date	datetime	Data do livro *Este campo é do tipo data.*
category	int	Categoria *Este campo é do tipo número inteiro.* *Chave estrangeira com a tabela t_categories com o campo id*
cover	image	Imagem representativa da capa do livro *Este campo é do tipo image onde se podem guardar ficheiros.*
file	image	Bytes do ficheiro carregado *Este campo é do tipo image onde se podem guardar ficheiros.*
file_name	varchar(100)	Nome do ficheiro carregado para posterior sugestão de gravação *Este campo é do tipo texto com um máximo de 100 carateres.*
book_resume	varchar(2000)	Resumo do livro *Este campo é do tipo texto com um máximo de 2000 carateres.* *Não suporta valores nulos.*

Tabela de Utilizadores

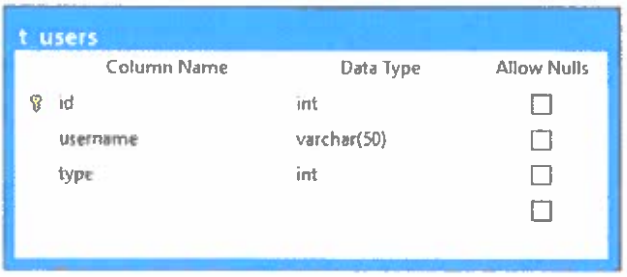

Figura 108 - Tabela de Utilizadores

Campo	*Tipo de dados*	*Descrição*
id	int	Identificação inequívoca do registo *Este campo é auto incremental do tipo número inteiro.*
username	varchar(50)	Nome do utilizador no formato domínio\utilizador *Este campo é do tipo texto com um máximo de 50 carateres.* *Não suporta valores nulos.*
type	int	Tipo de utilizador *Este campo é do tipo número inteiro.* *Chave estrangeira com a tabela t_usertypes com o campo id.* *Não suporta valores nulos.*

Tabela de Tipos de Utilizadores

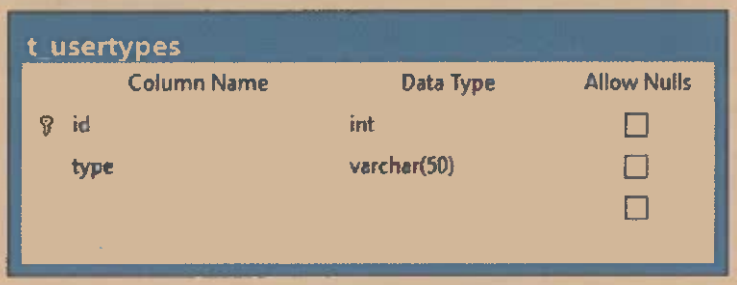

Figura 109 - Tabela de Tipos de Utilizadores

Campo	*Tipo de dados*	*Descrição*
id	int	Identificação inequívoca do registo *Este campo é auto incremental do tipo número inteiro.*
type	varchar(50)	Tipo de utilizador *Este campo é do tipo texto com um máximo de 50 carateres.* *Não suporta valores nulos.*

Tabela de Editoras

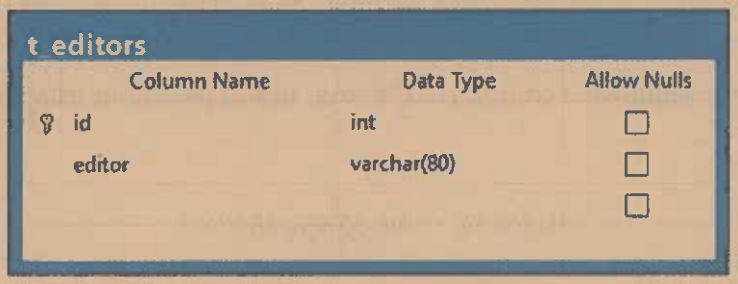

Figura 110 - Tabela de Editoras

Campo	*Tipo de dados*	*Descrição*
id	int	Identificação inequívoca do registo *Este campo é auto incremental do tipo número inteiro.*
editor	varchar(80)	Nome da editora *Este campo é do tipo texto com um máximo de 80 carateres.* *Não suporta valores nulos.*

Tabela de Categorias

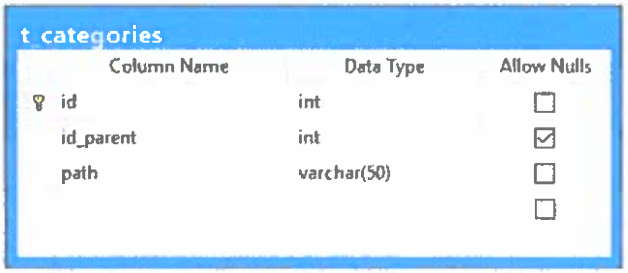

Figura 111 - Tabela de Categorias

Campo	*Tipo de dados*	*Descrição*
id	int	Identificação inequívoca do registo *Este campo é auto incremental do tipo número inteiro.*
id_parent	int	Identificação da categoria pai *Este campo é do tipo número inteiro.*
path	varchar(50)	Nome da categoria *Este campo é do tipo texto com um máximo de 50 carateres.* *Não suporta valores nulos.*

c) Visual Studio Community 2015 - Código fonte da aplicação em C# .NET

Neste ponto segue-se o código em C# .Net, usado para criar esta aplicação.

Classe - category_info.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;

namespace CeliSoft.Software
{
    /// <summary>
    ///     category_info
    ///     Serve para ser usada nos items da treeview das categorias
    /// </summary>
    public class category_info
    {
        #region Variables
        private int _id;
        private int _id_parent;
        private string _path;
```

```
        #endregion

        #region Properties
        public int id
        {
            get { return _id; }
            set { _id = value; }
        }
        public int id_parent
        {
            get { return _id_parent; }
            set { _id_parent = value; }
        }
        public string path
        {
            get { return _path; }
            set { _path = value; }
        }
        #endregion

        #region Constructor
        /// <summary>
        ///     Construtor
        /// </summary>
        /// <param name="id">id da categoria</param>
        /// <param name="id_parent">id da categoria 'pai'</param>
        /// <param name="path">categoria</param>
        public category_info(int id, int id_parent, string path)
        {
            _id = id;
            _id_parent = id_parent;
            _path = path;
        }
        #endregion
    }
}
```

Classe - ComboBoxItem.cs

```
namespace CeliSoft.Software
{
    /// <summary>
    ///     ComboBoxItem
    ///     Serve para ser usada nos items de uma combobox
    /// </summary>
    public class ComboBoxItem
    {
        #region Variables
        private int _ID;
        private string _Text;
        #endregion

        #region Properties
```

```
        public int ID
        {
            get { return _ID; }
            set { _ID = value; }
        }
        public string Text
        {
            get { return _Text; }
            set { _Text = value; }
        }
        #endregion

        #region Constructors
        /// <summary>
        ///     Construtor sem argumentos
        /// </summary>
        public ComboBoxItem()
        {

        }
        /// <summary>
        ///     Construtor com argumentos de inicio
        /// </summary>
        /// <param name="id">id do item</param>
        /// <param name="text">texto do item</param>
        public ComboBoxItem(int id, string text)
        {
            _ID = id;
            _Text = text;
        }
        #endregion

        #region Override Methods
        /// <summary>
        ///     Override do método ToString()
        /// </summary>
        /// <returns></returns>
        public override string ToString()
        {
            return _Text;
        }
        #endregion
    }
}
```

Classe - SelectedEventArgs.cs

```
namespace CeliSoft.Software
{
    /// <summary>
    ///     SelectedEventArgs
    ///     Serve para usar nos eventos dos items selecionados da
DataGridView
```

```
6       /// </summary>
7       public class SelectedEventArgs
8       {
9           #region Variables
10          private bool _Selected;
11          private int _SelectedIndex;
12          #endregion
13
14          #region Properties
15          public bool Selected
16          {
17              get { return _Selected; }
18              set { _Selected = value; }
19          }
20          public int SelectedIndex
21          {
22              get { return _SelectedIndex; }
23              set { _SelectedIndex = value; }
24          }
25          #endregion
26
27          #region Constructors
28          /// <summary>
29          ///     Construtor
30          /// </summary>
31          public SelectedEventArgs()
32          {
33
34          }
35          /// <summary>
36          ///     Construtor
37          /// </summary>
38          /// <param name="selected">Se o item da DataGridView está
selecionado</param>
39          /// <param name="selectedIndex">Qual é o índice do item
selecionado</param>
40          public SelectedEventArgs(bool selected, int selectedIndex)
41          {
42              this.Selected = selected;
43              this.SelectedIndex = selectedIndex;
44          }
45          #endregion
46      }
47  }
```

Formulário - frmMain.cs

```
1       using System;
2       using System.Drawing;
3       using System.Windows.Forms;
4       using RibbonLib.Controls;
5       using RibbonLib.Interop;
6       using System.Threading;
```

```
using System.ComponentModel;
using System.IO;

namespace CeliSoft.Software
{
    /// <summary>
    ///     frmMain
    ///     Janela principal da aplicação
    /// </summary>
    public partial class frmMain : Form
    {
        #region Window Initializing
        /// <summary>
        ///     Constructor
        /// </summary>
        public frmMain()
        {
            InitializeComponent();

            // Mostrar a form 'Splash Screen'
            frmSplash f_splash = new frmSplash();
            f_splash.ShowDialog(this);
            f_splash.Close();
            f_splash.Dispose();

            // Inicializar os controlos de contexto da Ribbon
            _ribbonBooksContextMenuStrip = new
RibbonContextMenuStrip(ribbon1,
(uint)RibbonMarkupCommands.cmdBooksContextMap);
            _ribbonCategoriesContextMenuStrip = new
RibbonContextMenuStrip(ribbon1,
(uint)RibbonMarkupCommands.cmdCategoriesContextMap);
            _ribbonEditorsContextMenuStrip = new
RibbonContextMenuStrip(ribbon1,
(uint)RibbonMarkupCommands.cmdEditorsContextMap);
        }
        #endregion

        #region Variables
        // Variáveis de acesso à base de dados com a tabela de
categorias
        DS_SQLAccessTableAdapters.CategoriesTableAdapter ta = new
DS_SQLAccessTableAdapters.CategoriesTableAdapter();
        DS_SQLAccess.CategoriesDataTable dt = null;
        #endregion

        #region Ribbon Controls Variables
        // Ribbon Commands
        private RibbonButton _cmdButtonNewBook;
        private RibbonButton _cmdButtonChangeBook;
        private RibbonButton _cmdButtonDeleteBook;
        private RibbonButton _cmdButtonUploadBook;
        private RibbonButton _cmdButtonDownloadBook;
```

```
52          private RibbonButton _cmdButtonSearch;
53          private RibbonButton _cmdButtonManageBooks;
54          private RibbonButton _cmdButtonManagePublishers;
55          private RibbonButton _cmdButtonManageUsers;
56          private RibbonButton _cmdButtonCloseTab;
57          private RibbonButton _cmdButtonCoverImage;
58          private RibbonToggleButton _cmdButtonViewCategories;
59          private RibbonToggleButton _cmdButtonViewDetails;
60          private RibbonButton _cmdButtonChangeBookCategory;
61          private RibbonToggleButton _cmdButtonViewStatusBar;
62          private RibbonButton _cmdButtonAddRootCategory;
63          private RibbonButton _cmdButtonAddChildCategory;
64          private RibbonButton _cmdButtonEditCategory;
65          private RibbonButton _cmdButtonRemoveCategory;
66          private RibbonButton _cmdButtonNewEditor;
67          private RibbonButton _cmdButtonChangeEditor;
68          private RibbonButton _cmdButtonDeleteEditor;
69          private RibbonButton _cmdButtonDownloadBooks;
70          private RibbonButton _cmdButtonNewUser;
71          private RibbonButton _cmdButtonChangeUser;
72          private RibbonButton _cmdButtonDeleteUser;
73          private RibbonButton _cmdButtonDeleteBooks;
74          private RibbonButton _cmdButtonExit;
75
76          // RibbonTab Groups
77          private RibbonTabGroup _cmdTabHome;
78          private RibbonTabGroup _cmdTabSelectedBook;
79          private RibbonTabGroup _cmdTabView;
80          private RibbonTabGroup _cmdTabSelectedEditor;
81          private RibbonTabGroup _cmdTabSelectedBooks;
82          private RibbonTabGroup _cmdTabSelectedCategory;
83          private RibbonTabGroup _cmdTabSelectedUser;
84
85          // Ribbon TabGroup Groups
86          private RibbonTabGroup _cmdTabGroupSelectedBookTools;
87          private RibbonTabGroup _cmdTabGroupSelectedEditorTools;
88          private RibbonTabGroup _cmdTabGroupSelectedBooksTools;
89          private RibbonTabGroup _cmdTabGroupSelectedCategoriesTools;
90          private RibbonTabGroup _cmdTabGroupSelectedUsersTools;
91
92          // Ribbon Group Groups
93          private RibbonGroup _cmdGroupBooks;
94          private RibbonGroup _cmdGroupDownUpLoad;
95          private RibbonGroup _cmdGroupCloseSearch;
96          private RibbonGroup _cmdGroupMaintenances;
97          private RibbonGroup _cmdGroupTabs;
98          private RibbonGroup _cmdGroupCover;
99          private RibbonGroup _cmdGroupVisualization;
100         private RibbonGroup _cmdGroupChangeCategory;
101         private RibbonGroup _cmdGroupEditors;
102         private RibbonGroup _cmdGroupAddButtons;
103         private RibbonGroup _cmdGroupUsers;
104
```

```
            // Ribbon Recent Items
            private RibbonRecentItems _cmdRecentItems;

            // Ribbon Context Menus
            public RibbonContextMenuStrip _ribbonBooksContextMenuStrip;
            public RibbonContextMenuStrip
_ribbonCategoriesContextMenuStrip;
            public RibbonContextMenuStrip _ribbonEditorsContextMenuStrip;
            #endregion

            #region Enums Code
            // Ribbon Commands IDs enum
            public enum RibbonMarkupCommands : uint
            {
                // Commands
                cmdButtonNewBook = 1001,
                cmdButtonChangeBook = 1002,
                cmdButtonDeleteBook = 1003,
                cmdButtonUploadBook = 1004,
                cmdButtonDownloadBook = 1005,
                cmdButtonSearch = 1006,
                cmdButtonManageBooks = 1007,
                cmdButtonManagePublishers = 1008,
                cmdButtonManageUsers = 1010,
                cmdButtonCloseTab = 1011,
                cmdButtonCoverImage = 1012,
                cmdButtonViewCategories = 1013,
                cmdButtonViewDetails = 1014,
                cmdButtonChangeBookCategory = 1015,
                cmdButtonViewStatusBar = 1016,

                cmdButtonAddRootCategory = 1017,
                cmdButtonAddChildCategory = 1018,
                cmdButtonEditCategory = 1019,
                cmdButtonRemoveCategory = 1020,
                cmdButtonNewEditor = 1021,
                cmdButtonChangeEditor = 1022,
                cmdButtonDeleteEditor = 1023,
                cmdButtonDownloadBooks = 1024,

                cmdButtonNewUser = 1028,
                cmdButtonChangeUser = 1029,
                cmdButtonDeleteUser = 1030,
                cmdButtonDeleteBooks = 1031,

                cmdButtonExit = 1099,

                // Tab Groups
                cmdTabHome = 2001,
                cmdTabSelectedBook = 2002,
                cmdTabView = 2003,
                cmdTabSelectedEditor = 2004,
                cmdTabSelectedBooks = 2005,
```

```
157                 cmdTabSelectedCategory = 2006,
158                 cmdTabSelectedUser = 2007,
159
160                 // TabGroup Groups
161                 cmdTabGroupSelectedBookTools = 2010,
162                 cmdTabGroupSelectedEditorTools = 2011,
163                 cmdTabGroupSelectedBooksTools = 2012,
164                 cmdTabGroupSelectedCategoriesTools = 2013,
165                 cmdTabGroupSelectedUsersTools = 2014,
166
167                 // Group Groups
168                 cmdGroupBooks = 3001,
169                 cmdGroupDownUpLoad = 303,
170                 //cmdGroupSearch = 3004,
171                 cmdGroupCloseSearch = 3005,
172                 cmdGroupMaintenances = 3006,
173                 cmdGroupTabs = 3007,
174                 cmdGroupCover = 3008,
175                 cmdGroupVisualization = 3009,
176                 cmdGroupChangeCategory = 3010,
177                 cmdGroupEditors = 3011,
178                 cmdGroupAddButtons = 3012,
179                 cmdGroupUsers= 3013,
180
181                 // Recent Items
182                 cmdRecentItems = 9000,
183
184                 cmdQAT = 10,
185                 cmdCustomizeQAT = 11,
186                 cmdBooksContextMap = 12,
187                 cmdCategoriesContextMap = 13,
188                 cmdEditorsContextMap = 14,
189                 cmdUsersContextMap = 15
190             }
191             // Enum de verificação de qual separador é qual
192             public enum enmTabsMode
193             {
194                 [Description("Books")]   Books,
195                 [Description("Search")]  Search,
196                 [Description("Editors")] Editors,
197                 [Description("Users")]   Users,
198             }
199             #endregion
200
201             #region Ribbon Controls Events
202             // Ribbon Commands Methods
203             private void _cmdButtonNewBook_ExecuteEvent(object sender,
RibbonLib.Controls.Events.ExecuteEventArgs e)
204             {
205                 string currentMode = string.Empty;
206                 ucBooks uc_Books = null;
207                 DataGridView dgvBooks = null;
208
```

```
                if (tabs.SelectedTab != null)
                {
                    currentMode = tabs.SelectedTab.Tag.ToString();
                    uc_Books =
((ucBooks)Utils.FindControl(tabs.SelectedTab, (currentMode ==
Utils.GetEnumDescription(enmTabsMode.Books) ? "uc_Books" : "uc_Search")));
                    dgvBooks =
((DataGridView)Utils.FindControl(tabs.SelectedTab, "dgvBooks"));
                }

                category_info cat_info =
((category_info)tvCategories.SelectedNode.Tag);

                // Criar um novo livro
                frmManBooks f_newBook = new frmManBooks(cat_info);
                f_newBook.Icon = Properties.Resources.books_pastel;
                f_newBook.Text = "Adicionar Livro";
                f_newBook.btnOK.Text = "Adicionar";
                if (f_newBook.ShowDialog() == DialogResult.OK)
                {
                    cat_info.id = f_newBook.idCategory;
                    cat_info.path = f_newBook.textCategory;
                    cat_info.id_parent = -1;

                    DS_SQLAccessTableAdapters.BooksTableAdapter ta = new
DS_SQLAccessTableAdapters.BooksTableAdapter();
                    DS_SQLAccess.BooksDataTable dt =
ta.GetData_Books(null, null, null, null, null, null, null, null, null, null);

                    DS_SQLAccess.BooksRow new_row = dt.NewBooksRow();

                    new_row.ref_code = f_newBook.txtReference.Text;
                    new_row.title = f_newBook.txtTitle.Text;
                    new_row.author = f_newBook.txtAuthor.Text;
                    if (f_newBook.cmbPublishers.SelectedIndex > 0)
                        new_row.editor =
((ComboBoxItem)f_newBook.cmbPublishers.SelectedItem).ID;
                    new_row.doc_type =
((ComboBoxItem)f_newBook.cmbDocTypes.SelectedItem).ID;
                    if (string.IsNullOrEmpty(f_newBook.txtDocSource.Text)
== false)
                        new_row.doc_source = f_newBook.txtDocSource.Text;
                    new_row.doc_date = f_newBook.dtpDocDate.Value;
                    new_row.category = cat_info.id;
                    new_row.book_resume = f_newBook.txtResume.Text;
                    dt.Rows.Add(new_row);

                    int ret = ta.Update(dt);
                    if (ret == 1)
                    {
                        // Recarregar os livros
                        if (uc_Books != null) uc_Books.LoadBooks();
                        if (dgvBooks != null)
Utils.SelectDataGridViewRow(dgvBooks, new_row.id);
```

```
253                 }
254             }
255         }
256         private void _cmdButtonChangeBook_ExecuteEvent(object sender,
RibbonLib.Controls.Events.ExecuteEventArgs e)
257         {
258             ucBooks uc_Books =
((ucBooks)Utils.FindControl(tabs.SelectedTab, "uc_Books"));
259             DataGridView dgvBooks =
((DataGridView)Utils.FindControl(tabs.SelectedTab, "dgvBooks"));
260             int id_book =
Convert.ToInt32(dgvBooks.SelectedRows[0].Cells[0].Value);
261
262             if (dgvBooks.SelectedRows.Count == 1)
263             {
264                 int id =
Convert.ToInt32(dgvBooks.SelectedRows[0].Cells[0].Value.ToString());
265                 int id_category =
Convert.ToInt32(dgvBooks.SelectedRows[0].Cells[8].Value.ToString());
266                 string category_fullpath = string.Empty;
267
268
DS_SQLAccessTableAdapters.Categories_GetCategoriesPathTableAdapter ta2 = new
DS_SQLAccessTableAdapters.Categories_GetCategoriesPathTableAdapter();
269                 DS_SQLAccess.Categories_GetCategoriesPathDataTable dt2
= ta2.GetData_GetCategoriesPath(id_category);
270
271                 DS_SQLAccess.Categories_GetCategoriesPathRow row2 =
((DS_SQLAccess.Categories_GetCategoriesPathRow)dt2.Rows[0]);
272
273                 if (row2 != null)
274                     category_fullpath = row2.fullpath;
275
276                 category_info cat_info =
Utils.GetCategoryByID(id_category);
277
278                 // Editar um livro
279                 frmManBooks f_editBook = new frmManBooks(cat_info);
280                 f_editBook.Icon = Properties.Resources.books_pastel;
281                 f_editBook.Text = "Editar Livro";
282
283                 f_editBook.txtReference.Text =
dgvBooks.SelectedRows[0].Cells[1].Value.ToString();
284                 f_editBook.txtTitle.Text =
dgvBooks.SelectedRows[0].Cells[2].Value.ToString();
285                 f_editBook.txtAuthor.Text =
dgvBooks.SelectedRows[0].Cells[3].Value.ToString();
286                 Utils.SelectComboboxItem(f_editBook.cmbDocTypes,
dgvBooks.SelectedRows[0].Cells[5].Value.ToString());
287                 Utils.SelectComboboxItem(f_editBook.cmbPublishers,
dgvBooks.SelectedRows[0].Cells[4].Value.ToString());
288                 f_editBook.txtDocSource.Text =
dgvBooks.SelectedRows[0].Cells[6].Value.ToString();
```

```
                    if (dgvBooks.SelectedRows[0].Cells[7].Value !=
DBNull.Value)
                        f_editBook.dtpDocDate.Value =
Convert.ToDateTime(dgvBooks.SelectedRows[0].Cells[7].Value.ToString());
                    f_editBook.txtResume.Text =
dgvBooks.SelectedRows[0].Cells[10].Value.ToString();
                    f_editBook.idCategory = id_category;
                    f_editBook.textCategory =
Utils.GetCategoryTextByID(f_editBook.idCategory);
                    f_editBook.category_fullpath = category_fullpath;
                    f_editBook.txtCategory.Text = f_editBook.textCategory;

                    f_editBook.btnOK.Text = "Editar";
                    if (f_editBook.ShowDialog() == DialogResult.OK)
                    {
                        DS_SQLAccessTableAdapters.BooksTableAdapter ta =
new DS_SQLAccessTableAdapters.BooksTableAdapter();
                        DS_SQLAccess.BooksDataTable dt =
ta.GetData_Books(id, null, null, null, null, null, null, null, null, null);

                        DS_SQLAccess.BooksRow edit_row =
((DS_SQLAccess.BooksRow)dt.Rows[0]);

                        edit_row.ref_code = f_editBook.txtReference.Text;
                        edit_row.title = f_editBook.txtTitle.Text;
                        edit_row.author = f_editBook.txtAuthor.Text;
                        if (f_editBook.cmbPublishers.SelectedIndex > 0)
                            edit_row.editor =
((ComboBoxItem)f_editBook.cmbPublishers.SelectedItem).ID;
                        edit_row.doc_type =
((ComboBoxItem)f_editBook.cmbDocTypes.SelectedItem).ID;
                        if
(string.IsNullOrEmpty(f_editBook.txtDocSource.Text) == false)
                            edit_row.doc_source =
f_editBook.txtDocSource.Text;
                        edit_row.doc_date = f_editBook.dtpDocDate.Value;
                        edit_row.category = f_editBook.idCategory;
                        edit_row.book_resume = f_editBook.txtResume.Text;

                        int ret = ta.Update(edit_row);
                        if (ret == 1)
                        {
                            if (uc_Books != null)
                            {
                                // Recarregar os livros
                                uc_Books.LoadBooks();

Utils.SelectDataGridViewRow(uc_Books.dgvBooks, id);
                            }
                        }
                    }
                }
            }
```

```
330         private void _cmdButtonDeleteBook_ExecuteEvent(object sender,
RibbonLib.Controls.Events.ExecuteEventArgs e)
331         {
332             ucBooks uc_Books = ((ucBooks)Utils.FindControl(tabs,
"uc_Books"));
333             DataGridView dgvBooks =
((DataGridView)Utils.FindControl(tabs, "dgvBooks"));
334
335             int count_selected_books = dgvBooks.SelectedRows.Count;
336
337             string esse = (count_selected_books == 1 ? string.Empty :
"s");
338             DialogResult resp = MessageBox.Show("Deseja remover o" +
esse + " livro" + esse + " selecionado" + esse + " ?", "Remover Livro" +
esse, MessageBoxButtons.YesNo, MessageBoxIcon.Question,
MessageBoxDefaultButton.Button2);
339
340             // Eliminar um livro
341             if (resp == DialogResult.Yes)
342             {
343                 DS_SQLAccessTableAdapters.BooksTableAdapter ta = new
DS_SQLAccessTableAdapters.BooksTableAdapter();
344                 DS_SQLAccess.BooksDataTable dt = null;
345
346                 bool deleted = false;
347                 foreach (DataGridViewRow dgv_row in
dgvBooks.SelectedRows)
348                 {
349                     int id_book =
Convert.ToInt32(dgv_row.Cells[0].Value);
350                     dt = ta.GetData_Books(id_book, null, null, null,
null, null, null, null, null, null);
351
352                     if (dt.Rows.Count == 1)
353                     {
354                         DS_SQLAccess.BooksRow del_row =
((DS_SQLAccess.BooksRow)dt.Rows[0]);
355                         del_row.Delete();
356
357                         int ret = ta.Update(del_row);
358                         if (ret == 1)
359                         {
360                             deleted = true;
361                         }
362                     }
363                 }
364
365                 // Recarregar os livros
366                 if (deleted) uc_Books.LoadBooks();
367             }
368         }
369         private void _cmdButtonUploadBook_ExecuteEvent(object sender,
RibbonLib.Controls.Events.ExecuteEventArgs e)
370         {
```

```
                string currentMode = string.Empty;
                ucBooks uc_Books = null;
                DataGridView dgvBooks = null;

                if (tabs.SelectedTab != null)
                {
                    currentMode = tabs.SelectedTab.Tag.ToString();
                    uc_Books =
((ucBooks)Utils.FindControl(tabs.SelectedTab, (currentMode ==
Utils.GetEnumDescription(enmTabsMode.Books) ? "uc_Books" : "uc_Search")));
                    dgvBooks =
((DataGridView)Utils.FindControl(tabs.SelectedTab, "dgvBooks"));
                }

                //Ler o ficheiro obtido do disco
                string filename = string.Empty;
                byte[] pdfbyte = Utils.ObterBytesFicheiro(null,
Utils.FiltrosSaveAs, "Obter ficheiro", out filename);

                //Se o ficheiro foi lido
                if (pdfbyte != null)
                {
                    //Obter o ID do livro selecionado
                    DataGridViewRow selected_row =
dgvBooks.SelectedRows[0];
                    int id = Convert.ToInt32(selected_row.Cells[0].Value);

                    //Ligar à BD por um TableAdapter
                    DS_SQLAccessTableAdapters.BooksTableAdapter tbta = new
DS_SQLAccessTableAdapters.BooksTableAdapter();
                    tbta.Connection.ConnectionString =
Utils.GetDBConnectionString();

                    //Carregar a tabela
                    DS_SQLAccess.BooksDataTable tbdt =
tbta.GetData_Books(id, null, null, null, null, null, null, null, null, null);

                    //Se encontrou o registo
                    if (tbdt != null && tbdt.Rows.Count > 0)
                    {
                        DS_SQLAccess.BooksRow row =
((DS_SQLAccess.BooksRow)tbdt.Rows[0]);
                        if (row != null)
                        {
                            //Atribuir o ficheiro PDF ao campo
                            row.file = pdfbyte;
                            row.file_name = Path.GetFileName(filename);

                            //Atualizar a base de dados
                            int ret = tbta.Update(row);
                            if (ret == 1)
                            {
                                //Disponibilizar o botão de Download do
PDF
```

```
                                _cmdButtonDownloadBook.Enabled = true;

                                uc_Books.LoadBooks();

Utils.SelectDataGridViewRow(uc_Books.dgvBooks, id);
                            }
                        }
                    }
                }

                EnableControls();

                //Libertar memória
                pdfbyte = null;
            }
            private void _cmdButtonDownloadBook_ExecuteEvent(object
sender, RibbonLib.Controls.Events.ExecuteEventArgs e)
            {
                string folder_path = string.Empty;

                FolderBrowserDialog bfd = new FolderBrowserDialog();
                bfd.Description = "Selecionar a pasta de destino dos
ficheiros";
                bfd.SelectedPath = Application.StartupPath;
                bfd.ShowNewFolderButton = true;

                if (bfd.ShowDialog() == DialogResult.OK)
                {
                    folder_path = bfd.SelectedPath;

                    ucBooks uc_Books = ((ucBooks)Utils.FindControl(tabs,
"uc_Books"));
                    DataGridView dgvBooks =
((DataGridView)Utils.FindControl(tabs, "dgvBooks"));

                    int files_saved = 0;
                    // Transferir todos os ficheiros de todos os livros
selecionados
                    foreach (DataGridViewRow rowSelected in
dgvBooks.SelectedRows)
                    {
                        int id_book =
Convert.ToInt32(rowSelected.Cells[0].Value);

                        DS_SQLAccessTableAdapters.BooksTableAdapter ta =
new DS_SQLAccessTableAdapters.BooksTableAdapter();
                        DS_SQLAccess.BooksDataTable dt =
ta.GetData_Books(id_book, null, null, null, null, null, null, null, null,
null);

                        byte[] file = null;
                        string filename = string.Empty;
                        DS_SQLAccess.BooksRow row_book =
((DS_SQLAccess.BooksRow)dt.Rows[0]);
```

```
                        file = row_book.file;

                        Guid guid = Guid.NewGuid();
                        if (row_book.Isfile_nameNull())
                            filename = guid.ToString().Replace("{",
string.Empty).Replace("}", string.Empty).Replace("-", string.Empty) + ".ext";
                        else
                            filename = row_book.file_name;

                        Utils.ConverterBytesEmFicheiro(folder_path + @"\"
+ filename, file);
                        files_saved++;
                    }

                    MessageBox.Show("Ficheiros guardados: " +
files_saved.ToString(), "Ficheiros guardados", MessageBoxButtons.OK,
(files_saved == 0 ? MessageBoxIcon.Exclamation :
MessageBoxIcon.Information));
                }
            }
            private void _cmdButtonManageBooks_ExecuteEvent(object sender,
RibbonLib.Controls.Events.ExecuteEventArgs e)
            {
                OpenBooksPage(enmTabsMode.Books);
            }
            private void _cmdButtonManagePublishers_ExecuteEvent(object
sender, RibbonLib.Controls.Events.ExecuteEventArgs e)
            {
                OpenEditorsPage();
            }
            private void _cmdButtonCloseTab_ExecuteEvent(object sender,
RibbonLib.Controls.Events.ExecuteEventArgs e)
            {
                UnSelectContextTab();

                if (tabs.TabPages.Count > 0)
                {
                    string tagCurrentTab = (tabs.SelectedTab.Tag != null ?
tabs.SelectedTab.Tag.ToString() : string.Empty);

                    // Fechar separador atual
                    Utils.RemovingTab = tagCurrentTab;

tabs.TabPages.Remove(tabs.TabPages[tabs.SelectedIndex]);
                    TabClosed(tabs, null);
                }
            }
            private void _cmdButtonCoverImage_ExecuteEvent(object sender,
RibbonLib.Controls.Events.ExecuteEventArgs e)
            {
                // Obter uma imagem para a capa do livro

                DataGridViewRow selected_row =
Utils.GetSelectedRow(tabs.TabPages);
```

```
499                 string filename = null;
500                 byte[] bytes = Utils.ObterBytesFicheiro(null, "Ficheiros
de Imagens (*.png; *.jpg; *.bmp)|*.png; *.jpg; *.bmp|All Files (*.*|*.*)",
"Selecionar Capa do Livro '" + selected_row.Cells[2].Value.ToString() + "'",
out filename);
501
502                 if (bytes != null && bytes.Length > 0)
503                 {
504                     int id = Convert.ToInt32(selected_row.Cells[0].Value);
505                     DS_SQLAccessTableAdapters.BooksTableAdapter ta = new
DS_SQLAccessTableAdapters.BooksTableAdapter();
506                     DS_SQLAccess.BooksDataTable dt = ta.GetData_Books(id,
null, null, null, null, null, null, null, null, null);
507
508                     DS_SQLAccess.BooksRow row_edit =
((DS_SQLAccess.BooksRow)dt.Rows[0]);
509
510                     if (row_edit != null)
511                     {
512                         row_edit.cover = bytes;
513
514                         int ret = ta.Update(row_edit);
515                         if (ret == 1)
516                         {
517                             ucBooks uc_sb =
(ucBooks)Utils.FindControl((Control)tabs, "uc_Books");
518                             if (uc_sb != null)
519                             {
520                                 uc_sb.category =
((category_info)tvCategories.SelectedNode.Tag).id;
521                                 uc_sb.LoadBooks();
522
Utils.SelectDataGridViewRow(uc_sb.dgvBooks, id);
523                             }
524                         }
525                     }
526                 }
527             }
528             private void _cmdButtonViewCategories_ExecuteEvent(object
sender, RibbonLib.Controls.Events.ExecuteEventArgs e)
529             {
530                 // Mostrar ou não as categorias
531                 splitContainerMain.Panel1Collapsed =
!_cmdButtonViewCategories.BooleanValue;
532             }
533             private void _cmdButtonViewDetails_ExecuteEvent(object sender,
RibbonLib.Controls.Events.ExecuteEventArgs e)
534             {
535                 if (tabs.SelectedTab != null)
536                 {
537                     string currentMode = tabs.SelectedTab.Tag.ToString();
538                     ucBooks uc_Books =
((ucBooks)Utils.FindControl(tabs.SelectedTab, (currentMode ==
Utils.GetEnumDescription(enmTabsMode.Books) ? "uc_Books" : "uc_Search")));
```

```
                    DataGridView dgvBooks =
((DataGridView)Utils.FindControl(tabs.SelectedTab, "dgvBooks"));

                    if (uc_Books != null)
                    {
                        // Mostrar ou não as os detalhes do livro
selecionado
                        uc_Books.pnlDetails.Visible =
_cmdButtonViewDetails.BooleanValue;
                        uc_Books.pnlDetails.Refresh();
                    }
                }
            }
            private void _cmdButtonChangeBookCategory_ExecuteEvent(object
sender, RibbonLib.Controls.Events.ExecuteEventArgs e)
            {
                //Alterar a categoria do(s) livro(s)

                ribbon1.Cursor = Cursors.WaitCursor;
                Application.DoEvents();

                Thread.Sleep(200);

                category_info cat_info =
((category_info)tvCategories.SelectedNode.Tag);

                string category_fullpath = string.Empty;


DS_SQLAccessTableAdapters.Categories_GetCategoriesPathTableAdapter ta2 = new
DS_SQLAccessTableAdapters.Categories_GetCategoriesPathTableAdapter();
                DS_SQLAccess.Categories_GetCategoriesPathDataTable dt2 =
ta2.GetData_GetCategoriesPath(cat_info.id);

                DS_SQLAccess.Categories_GetCategoriesPathRow row2 =
((DS_SQLAccess.Categories_GetCategoriesPathRow)dt2.Rows[0]);

                if (row2 != null)
                    category_fullpath = row2.fullpath;

                string currentMode = tabs.SelectedTab.Tag.ToString();
                ucBooks uc_Books =
((ucBooks)Utils.FindControl(tabs.SelectedTab, (currentMode ==
Utils.GetEnumDescription(enmTabsMode.Books) ? "uc_Books" : "uc_Search")));
                DataGridView dgvBooks =
((DataGridView)Utils.FindControl(tabs.SelectedTab, "dgvBooks"));

                frmSelectCategory f_select = new
frmSelectCategory(cat_info.id);
                f_select.category_path = category_fullpath;
                f_select.f_main = this;

                if (f_select.ShowDialog() == DialogResult.OK)
                {
```

```
580                     int list_changed = 0;
581                     int id_book_changed = 0;
582                     foreach (DataGridViewRow rowSelected in
dgvBooks.SelectedRows)
583                     {
584                         int id_book =
Convert.ToInt32(rowSelected.Cells[0].Value);
585
586                         DS_SQLAccessTableAdapters.BooksTableAdapter ta =
new DS_SQLAccessTableAdapters.BooksTableAdapter();
587                         DS_SQLAccess.BooksDataTable dt =
ta.GetData_Books(id_book, null, null, null, null, null, null, null, null,
null);
588
589                         int idCategory = f_select.selected_id;
590                         string textCategory = f_select.selected_category;
591
592                         DS_SQLAccess.BooksRow row_category =
((DS_SQLAccess.BooksRow)dt.Rows[0]);
593                         row_category.category = idCategory;
594
595                         int ret = ta.Update(row_category);
596                         if (ret == 1)
597                         {
598                             list_changed++;
599                             id_book_changed = id_book;
600                         }
601                     }
602
603                     if (list_changed > 0)
604                     {
605                         // Recarregar os livros
606                         uc_Books.LoadBooks();
607
608                         if (list_changed == 1)
609                             Utils.SelectDataGridViewRow(dgvBooks,
id_book_changed);
610                     }
611                 }
612
613                 ribbon1.Cursor = Cursors.Default;
614                 Application.DoEvents();
615             }
616             private void _cmdButtonViewStatusBar_ExecuteEvent(object
sender, RibbonLib.Controls.Events.ExecuteEventArgs e)
617             {
618                 // Mostrar ou não a barra de estado
619                 statusStrip1.Visible =
_cmdButtonViewStatusBar.BooleanValue;
620             }
621             private void _cmdButtonNewEditor_ExecuteEvent(object sender,
RibbonLib.Controls.Events.ExecuteEventArgs e)
622             {
```

```
                ucEditors uc_Editors = ((ucEditors)Utils.FindControl(tabs,
"uc_Editors"));
                DataGridView dgvEditors =
((DataGridView)Utils.FindControl(tabs, "dgvEditors"));

                // Criar uma nova editora
                frmManEditors f_newEditor = new frmManEditors();
                f_newEditor.Text = "Adicionar Editora";
                f_newEditor.btnOK.Text = "Adicionar";
                if (f_newEditor.ShowDialog() == DialogResult.OK)
                {
                    DS_SQLAccessTableAdapters.EditorsTableAdapter ta = new
DS_SQLAccessTableAdapters.EditorsTableAdapter();
                    DS_SQLAccess.EditorsDataTable dt =
ta.GetData_Editors(null, null);

                    DS_SQLAccess.EditorsRow new_row = dt.NewEditorsRow();

                    new_row.editor = f_newEditor.txtEditor.Text;
                    dt.Rows.Add(new_row);

                    int ret = ta.Update(dt);
                    if (ret == 1)
                    {
                        // Recarregar as editoras
                        if (uc_Editors != null) uc_Editors.LoadEditors();
                        if (dgvEditors != null)
Utils.SelectDataGridViewRow(dgvEditors, new_row.id);
                    }
                }
            }
            private void _cmdButtonChangeEditor_ExecuteEvent(object
sender, RibbonLib.Controls.Events.ExecuteEventArgs e)
            {
                ucEditors uc_Editors = ((ucEditors)Utils.FindControl(tabs,
"uc_Editors"));
                DataGridView dgvEditors =
((DataGridView)Utils.FindControl(tabs, "dgvEditors"));
                int id_Editor =
Convert.ToInt32(dgvEditors.SelectedRows[0].Cells[0].Value);

                if (dgvEditors.SelectedRows.Count == 1)
                {
                    int id =
Convert.ToInt32(dgvEditors.SelectedRows[0].Cells[0].Value.ToString());

                    // Editar uma categoria selecionada
                    frmManEditors f_editEditor = new frmManEditors();
                    f_editEditor.txtID.Text = id_Editor.ToString();
                    f_editEditor.Text = "Editar Editora";

                    f_editEditor.txtEditor.Text =
dgvEditors.SelectedRows[0].Cells[1].Value.ToString();
```

```
                    f_editEditor.btnOK.Text = "Editar";
                    if (f_editEditor.ShowDialog() == DialogResult.OK)
                    {
                        DS_SQLAccessTableAdapters.EditorsTableAdapter ta =
new DS_SQLAccessTableAdapters.EditorsTableAdapter();
                        DS_SQLAccess.EditorsDataTable dt =
ta.GetData_Editors(id, null);

                        DS_SQLAccess.EditorsRow edit_row =
((DS_SQLAccess.EditorsRow)dt.Rows[0]);

                        edit_row.editor = f_editEditor.txtEditor.Text;

                        int ret = ta.Update(edit_row);
                        if (ret == 1)
                        {
                            // Recarregar as editoras
                            if (uc_Editors != null)
                                uc_Editors.LoadEditors();
                        }
                    }
                }
            }
            private void _cmdButtonDeleteEditor_ExecuteEvent(object
sender, RibbonLib.Controls.Events.ExecuteEventArgs e)
            {
                ucEditors uc_Editors = ((ucEditors)Utils.FindControl(tabs,
"uc_Editors"));
                DataGridView dgvEditors =
((DataGridView)Utils.FindControl(tabs, "dgvEditors"));

                int count_selected_editors =
dgvEditors.SelectedRows.Count;

                DialogResult resp = MessageBox.Show("Deseja remover a
editora selecionada ?", "Remover Editora", MessageBoxButtons.YesNo,
MessageBoxIcon.Question, MessageBoxDefaultButton.Button2);

                // Eliminar a editora selecionada
                if (resp == DialogResult.Yes)
                {
                    DS_SQLAccessTableAdapters.EditorsTableAdapter ta = new
DS_SQLAccessTableAdapters.EditorsTableAdapter();
                    DS_SQLAccess.EditorsDataTable dt = null;

                    int id_editor =
Convert.ToInt32(dgvEditors.SelectedRows[0].Cells[0].Value);
                    dt = ta.GetData_Editors(id_editor, null);

                    if (dt.Rows.Count == 1)
                    {
                        DS_SQLAccess.EditorsRow del_row =
((DS_SQLAccess.EditorsRow)dt.Rows[0]);
                        del_row.Delete();
```

```
                    int ret = ta.Update(del_row);
                    if (ret == 1)
                    {
                        // Recarregar as editoras
                        if (uc_Editors != null)
                            uc_Editors.LoadEditors();
                    }
                }
            }
        }
        private void _cmdButtonManageUsers_ExecuteEvent(object sender,
RibbonLib.Controls.Events.ExecuteEventArgs e)
        {
            OpenUsersPage();
        }
        private void _cmdButtonNewUser_ExecuteEvent(object sender,
RibbonLib.Controls.Events.ExecuteEventArgs e)
        {
            ucUsers uc_Users = ((ucUsers)Utils.FindControl(tabs,
"uc_Users"));
            DataGridView dgvUsers =
((DataGridView)Utils.FindControl(tabs, "dgvUsers"));

            // Criar um novo utilizador
            frmManUsers f_newUser = new frmManUsers();
            f_newUser.Text = "Adicionar Utilizador";
            f_newUser.btnOK.Text = "Adicionar";
            if (f_newUser.ShowDialog() == DialogResult.OK)
            {
                DS_SQLAccessTableAdapters.UsersTableAdapter ta = new
DS_SQLAccessTableAdapters.UsersTableAdapter();
                DS_SQLAccess.UsersDataTable dt =
ta.GetData_Users(null, null, null);

                DS_SQLAccess.UsersRow new_row = dt.NewUsersRow();

                new_row.username = f_newUser.txtUsername.Text;
                new_row.type =
((ComboBoxItem)f_newUser.cmbUserType.SelectedItem).ID;
                dt.Rows.Add(new_row);

                int ret = ta.Update(dt);
                if (ret == 1)
                {
                    // Recarregar os utilizadores
                    if (uc_Users != null) uc_Users.LoadUsers();
                    if (dgvUsers != null)
Utils.SelectDataGridViewRow(dgvUsers, new_row.id);
                }
            }
        }
```

```
752             private void _cmdButtonChangeUser_ExecuteEvent(object sender,
RibbonLib.Controls.Events.ExecuteEventArgs e)
753             {
754                 ucUsers uc_Users = ((ucUsers)Utils.FindControl(tabs,
"uc_Users"));
755                 DataGridView dgvUsers =
((DataGridView)Utils.FindControl(tabs, "dgvUsers"));
756                 int id_User =
Convert.ToInt32(dgvUsers.SelectedRows[0].Cells[0].Value);
757
758                 if (dgvUsers.SelectedRows.Count == 1)
759                 {
760                     int id =
Convert.ToInt32(dgvUsers.SelectedRows[0].Cells[0].Value.ToString());
761
762                     // Editar o utilizador selecionado
763                     frmManUsers f_editUser = new frmManUsers();
764                     f_editUser.txtID.Text = id_User.ToString();
765                     f_editUser.Text = "Editar Utilizador";
766
767                     f_editUser.txtUsername.Text =
dgvUsers.SelectedRows[0].Cells[1].Value.ToString();
768                     Utils.SelectComboboxItem(f_editUser.cmbUserType,
dgvUsers.SelectedRows[0].Cells[2].Value.ToString());
769
770                     f_editUser.btnOK.Text = "Editar";
771                     if (f_editUser.ShowDialog() == DialogResult.OK)
772                     {
773                         DS_SQLAccessTableAdapters.UsersTableAdapter ta =
new DS_SQLAccessTableAdapters.UsersTableAdapter();
774                         DS_SQLAccess.UsersDataTable dt =
ta.GetData_Users(id, null, null);
775
776                         DS_SQLAccess.UsersRow edit_row =
((DS_SQLAccess.UsersRow)dt.Rows[0]);
777
778                         edit_row.username = f_editUser.txtUsername.Text;
779
780                         int ret = ta.Update(edit_row);
781                         if (ret == 1)
782                         {
783                             // Recarregar os utilizadores
784                             if (uc_Users != null)
785                                 uc_Users.LoadUsers();
786                         }
787                     }
788                 }
789             }
790             private void _cmdButtonDeleteUser_ExecuteEvent(object sender,
RibbonLib.Controls.Events.ExecuteEventArgs e)
791             {
792                 ucUsers uc_Users = ((ucUsers)Utils.FindControl(tabs,
"uc_Users"));
```

```
                DataGridView dgvUsers =
((DataGridView)Utils.FindControl(tabs, "dgvUsers"));

                int count_selected_Users = dgvUsers.SelectedRows.Count;

                DialogResult resp = MessageBox.Show("Deseja remover o
utilizador selecionado ?", "Remover Utilizador", MessageBoxButtons.YesNo,
MessageBoxIcon.Question, MessageBoxDefaultButton.Button2);

                // Eliminar o utilizador selecionado
                if (resp == DialogResult.Yes)
                {
                    DS_SQLAccessTableAdapters.UsersTableAdapter ta = new
DS_SQLAccessTableAdapters.UsersTableAdapter();
                    DS_SQLAccess.UsersDataTable dt = null;

                    int id_user =
Convert.ToInt32(dgvUsers.SelectedRows[0].Cells[0].Value);
                    dt = ta.GetData_Users(id_user, null, null);

                    if (dt.Rows.Count == 1)
                    {
                        DS_SQLAccess.UsersRow del_row =
((DS_SQLAccess.UsersRow)dt.Rows[0]);
                        del_row.Delete();

                        int ret = ta.Update(del_row);
                        if (ret == 1)
                        {
                            // Recarregar os utilizadores
                            if (uc_Users != null)
                                uc_Users.LoadUsers();
                        }
                    }
                }
            }
            private void _cmdButtonDownloadBooks_ExecuteEvent(object
sender, RibbonLib.Controls.Events.ExecuteEventArgs e)
            {

_cmdButtonDownloadBook_ExecuteEvent(_cmdButtonDownloadBook, e);
            }
            private void _cmdButtonDeleteBooks_ExecuteEvent(object sender,
RibbonLib.Controls.Events.ExecuteEventArgs e)
            {
                _cmdButtonDeleteBook_ExecuteEvent(_cmdButtonDeleteBooks,
e);
            }
            private void _cmdButtonAddRootCategory_ExecuteEvent(object
sender, RibbonLib.Controls.Events.ExecuteEventArgs e)
            {
                // Adicionar categoria
                frmManCategories f_manCateg = new frmManCategories();
                f_manCateg.Text = "Adicionar Categoria";
```

```
836                 f_manCateg.btnOK.Text = "Adicionar";
837                 if (f_manCateg.ShowDialog() == DialogResult.OK)
838                 {
839                     DS_SQLAccessTableAdapters.CategoriesTableAdapter ta =
new DS_SQLAccessTableAdapters.CategoriesTableAdapter();
840                     DS_SQLAccess.CategoriesDataTable dt =
ta.GetData_Categories(null, -1, null);
841
842                     DS_SQLAccess.CategoriesRow new_row =
dt.NewCategoriesRow();
843
844                     new_row.id_parent = -1;
845                     new_row.path = f_manCateg.txtCategory.Text;
846                     dt.Rows.Add(new_row);
847
848                     int ret = ta.Update(dt);
849                     if (ret == 1)
850                     {
851                         // Recarregar Categorias
852                         string category_fullpath =
tvCategories.SelectedNode.FullPath;
853                         LoadCategories(tvCategories, -1);
854                         Utils.ExpandTreeByFullPath(this, tvCategories,
category_fullpath);
855                     }
856                 }
857             }
858             private void _cmdButtonAddChildCategory_ExecuteEvent(object
sender, RibbonLib.Controls.Events.ExecuteEventArgs e)
859             {
860                 // Adicionar subcategoria
861                 frmManCategories f_manCateg = new frmManCategories();
862                 category_info cat_info =
((category_info)tvCategories.SelectedNode.Tag);
863                 f_manCateg.Text = "Adicionar Sub-Categoria";
864                 f_manCateg.btnOK.Text = "Adicionar";
865                 if (f_manCateg.ShowDialog() == DialogResult.OK)
866                 {
867                     DS_SQLAccessTableAdapters.CategoriesTableAdapter ta =
new DS_SQLAccessTableAdapters.CategoriesTableAdapter();
868                     DS_SQLAccess.CategoriesDataTable dt =
ta.GetData_Categories(null, cat_info.id, null);
869
870                     DS_SQLAccess.CategoriesRow new_row =
dt.NewCategoriesRow();
871
872                     new_row.id_parent = cat_info.id;
873                     new_row.path = f_manCateg.txtCategory.Text;
874                     dt.Rows.Add(new_row);
875
876                     int ret = ta.Update(dt);
877                     if (ret == 1)
878                     {
879                         // Recarregar Categorias
```

```
                        string category_fullpath =
tvCategories.SelectedNode.FullPath;
                        LoadCategories(tvCategories, -1);
                        Utils.ExpandTreeByFullPath(this, tvCategories,
category_fullpath);
                    }
                }
            }
            private void _cmdButtonEditCategory_ExecuteEvent(object
sender, RibbonLib.Controls.Events.ExecuteEventArgs e)
            {
                // Editar (sub)categoria
                frmManCategories f_manCateg = new frmManCategories();
                category_info cat_info =
((category_info)tvCategories.SelectedNode.Tag);
                f_manCateg.txtCategory.Text = cat_info.path;
                f_manCateg.Text = "Editar " + (cat_info.id_parent == -1 ?
"" : "Sub-") + "Categoria";
                f_manCateg.btnOK.Text = "Editar";
                if (f_manCateg.ShowDialog() == DialogResult.OK)
                {
                    DS_SQLAccessTableAdapters.CategoriesTableAdapter ta =
new DS_SQLAccessTableAdapters.CategoriesTableAdapter();
                    DS_SQLAccess.CategoriesDataTable dt =
ta.GetData_Categories(cat_info.id, cat_info.id_parent, null);

                    DS_SQLAccess.CategoriesRow edit_row =
((DS_SQLAccess.CategoriesRow)dt.Rows[0]);

                    edit_row.id = cat_info.id;
                    edit_row.id_parent = cat_info.id_parent;
                    edit_row.path = f_manCateg.txtCategory.Text;

                    int ret = ta.Update(edit_row);
                    if (ret == 1)
                    {
                        // Recarregar Categorias
                        string category_fullpath =
tvCategories.SelectedNode.FullPath;
                        LoadCategories(tvCategories, -1);
                        Utils.ExpandTreeByFullPath(this, tvCategories,
category_fullpath);
                    }
                }
            }
            private void _cmdButtonRemoveCategory_ExecuteEvent(object
sender, RibbonLib.Controls.Events.ExecuteEventArgs e)
            {
                string currentMode = tabs.SelectedTab.Tag.ToString();
                ucBooks uc_Books =
((ucBooks)Utils.FindControl(tabs.SelectedTab, (currentMode ==
Utils.GetEnumDescription(enmTabsMode.Books) ? "uc_Books" : "uc_Search")));

```

```
                category_info cat_info =
((category_info)tvCategories.SelectedNode.Tag);

                DialogResult resp = MessageBox.Show("Deseja remover a
categoria selecionada ?\r\n\r\nATENÇÃO: Os livros que constem nestas
categorias (e sub-categorias) serão associados à categoria (Sem categoria)",
"Remover Categoria", MessageBoxButtons.YesNo, MessageBoxIcon.Question,
MessageBoxDefaultButton.Button2);

                // Eliminar a categoria selecionada
                if (resp == DialogResult.Yes)
                {
                    DS_SQLAccessTableAdapters.CategoriesTableAdapter ta =
new DS_SQLAccessTableAdapters.CategoriesTableAdapter();
                    DS_SQLAccess.CategoriesDataTable dt =
ta.GetData_Categories(cat_info.id, null, null);

                    DS_SQLAccess.CategoriesRow delete_row =
((DS_SQLAccess.CategoriesRow)dt.Rows[0]);

                    delete_row.Delete();

                    int ret = ta.Update(delete_row);
                    if (ret == 1)
                    {
                        // Recarregar Categorias
                        LoadCategories(tvCategories, -1);

                        uc_Books =
((ucBooks)Utils.FindControl(tabs.SelectedTab, (currentMode ==
Utils.GetEnumDescription(enmTabsMode.Books) ? "uc_Books" : "uc_Search")));
                        bool hasItemSelected =
uc_Books.dgvBooks.SelectedRows.Count > 0;
                        uc_Books.LoadBooks();
                        if (!hasItemSelected)
uc_Books.dgvBooks.ClearSelection();
                    }
                }
            }
            private void _cmdButtonSearch_ExecuteEvent(object sender,
RibbonLib.Controls.Events.ExecuteEventArgs e)
            {
                OpenBooksSearchPage();
            }
            private void _cmdButtonExit_ExecuteEvent(object sender,
RibbonLib.Controls.Events.ExecuteEventArgs e)
            {
                // Close form asynchronously since we are in a ribbon
event
                // handler, so the ribbon is still in use, and calling
Close
                // will eventually call _ribbon.DestroyFramework(), which
is
                // a big no-no, if you still use the ribbon.
```

```
                this.BeginInvoke(new MethodInvoker(this.Close));
            }
            #endregion

            #region Window Methods
            /// <summary>
            ///     Window Load Method
            /// </summary>
            /// <param name="sender"></param>
            /// <param name="e"></param>
            private void frmMain_Load(object sender, EventArgs e)
            {
                //Atribuir a versão à barra de título da aplicação
                this.Text += " - Versão " + Utils.ProductVersion;

                // Iniciar sessão
                LoginWithCurrentWindowsUser();

                // Initialize Ribbon Variables
                InitializeRibbonVariables();

                // Disponibilizar ou não os controlos
                EnableControls();

                // Load categories
                LoadCategories(tvCategories, -1);

                if (tvCategories.Nodes.Count > 0)
                    tvCategories.SelectedNode = tvCategories.Nodes[0];
            }
            #endregion

            #region Methods
            /// <summary>
            ///     Código para disponibilizar ou não controlos com base
no flluxo ou tipo de utilizador
            /// </summary>
            private void EnableControls()
            {
                ucBooks uc_Books = null;
                DataGridView dgvBooks = null;

                if (tabs.TabPages.Count > 0)
                {
                    TabPage tabPage = tabs.SelectedTab;
                    string currentMode = string.Empty;
                    if (tabPage != null)
                    {
                        currentMode = tabs.SelectedTab.Tag.ToString();
                        uc_Books =
((ucBooks)Utils.FindControl(tabs.SelectedTab, (currentMode ==
Utils.GetEnumDescription(enmTabsMode.Books) ? "uc_Books" : "uc_Search")));
```

```
1006                    dgvBooks =
((DataGridView)Utils.FindControl(tabs.SelectedTab, "dgvBooks"));
1007                }
1008            }
1009
1010            bool books_or_search = false;
1011            if (uc_Books != null)
1012            {
1013                books_or_search = true;
1014            }
1015
1016            if (_cmdButtonViewDetails != null)
1017            {
1018                _cmdButtonViewDetails.Enabled = books_or_search;
1019                if (tvCategories.SelectedNode != null &&
typeof(category_info).IsAssignableFrom(tvCategories.SelectedNode.Tag.GetType(
)))
1020                {
1021                    _cmdButtonAddChildCategory.Enabled =
((category_info)tvCategories.SelectedNode.Tag).id != 1;
1022                    _cmdButtonEditCategory.Enabled =
((category_info)tvCategories.SelectedNode.Tag).id != 1;
1023                    _cmdButtonRemoveCategory.Enabled =
((category_info)tvCategories.SelectedNode.Tag).id != 1;
1024                }
1025                else
1026                {
1027                    _cmdButtonAddChildCategory.Enabled = false;
1028                    _cmdButtonEditCategory.Enabled = false;
1029                    _cmdButtonRemoveCategory.Enabled = false;
1030                }
1031            }
1032
1033            // Se fôr apenas utilizador
1034            if (Utils.CurrentUserType == 2)
1035            {
1036                _cmdButtonNewBook.Enabled = false;
1037                _cmdButtonChangeBook.Enabled = false;
1038                _cmdButtonDeleteBook.Enabled = false;
1039                _cmdButtonUploadBook.Enabled = false;
1040                _cmdButtonManagePublishers.Enabled = false;
1041                _cmdButtonManageUsers.Enabled = false;
1042                _cmdButtonCoverImage.Enabled = false;
1043                _cmdButtonChangeBookCategory.Enabled = false;
1044                _cmdButtonAddRootCategory.Enabled = false;
1045                _cmdButtonAddChildCategory.Enabled = false;
1046                _cmdButtonEditCategory.Enabled = false;
1047                _cmdButtonRemoveCategory.Enabled = false;
1048                _cmdButtonNewEditor.Enabled = false;
1049                _cmdButtonChangeEditor.Enabled = false;
1050                _cmdButtonDeleteEditor.Enabled = false;
1051                _cmdButtonNewUser.Enabled = false;
1052                _cmdButtonChangeUser.Enabled = false;
```

```
                _cmdButtonDeleteUser.Enabled = false;
                _cmdButtonDeleteBooks.Enabled = false;

                _ribbonCategoriesContextMenuStrip.Enabled = false;
                _ribbonEditorsContextMenuStrip.Enabled = false;
            }

            DS_SQLAccessTableAdapters.BooksTableAdapter ta = new
DS_SQLAccessTableAdapters.BooksTableAdapter();
            DS_SQLAccess.BooksDataTable dt = ta.GetData_Books(null,
null, null, null, null, null, null, null, null, null);

            DS_SQLAccessTableAdapters.UsersTableAdapter ta2 = new
DS_SQLAccessTableAdapters.UsersTableAdapter();
            DS_SQLAccess.UsersDataTable dt2 = ta2.GetData_Users(null,
null, null);

            // Procura de quantos livros e utilizadores existem na
base de dados
            int book_count = dt.Rows.Count;
            int user_count = dt2.Rows.Count;

            tssMessages.Text = string.Format("Livros encontrados: {0}
/ Utilizadores registados: {1}", book_count, user_count);
        }

        /// <summary>
        ///     Inicialização das variáveis da Ribbon
        /// </summary>
        private void InitializeRibbonVariables()
        {
            // Commands
            _cmdButtonNewBook = new RibbonButton(ribbon1,
(uint)RibbonMarkupCommands.cmdButtonNewBook);
            _cmdButtonNewBook.ExecuteEvent +=
_cmdButtonNewBook_ExecuteEvent;
            _cmdButtonChangeBook = new RibbonButton(ribbon1,
(uint)RibbonMarkupCommands.cmdButtonChangeBook);
            _cmdButtonChangeBook.ExecuteEvent +=
_cmdButtonChangeBook_ExecuteEvent;
            _cmdButtonDeleteBook = new RibbonButton(ribbon1,
(uint)RibbonMarkupCommands.cmdButtonDeleteBook);
            _cmdButtonDeleteBook.ExecuteEvent +=
_cmdButtonDeleteBook_ExecuteEvent;
            _cmdButtonUploadBook = new RibbonButton(ribbon1,
(uint)RibbonMarkupCommands.cmdButtonUploadBook);
            _cmdButtonUploadBook.ExecuteEvent +=
_cmdButtonUploadBook_ExecuteEvent;
            _cmdButtonDownloadBook = new RibbonButton(ribbon1,
(uint)RibbonMarkupCommands.cmdButtonDownloadBook);
            _cmdButtonDownloadBook.ExecuteEvent +=
_cmdButtonDownloadBook_ExecuteEvent;
            _cmdButtonDownloadBook.Enabled = false;
```

```
1090                _cmdButtonSearch = new RibbonButton(ribbon1,
(uint)RibbonMarkupCommands.cmdButtonSearch);
1091                _cmdButtonSearch.ExecuteEvent +=
_cmdButtonSearch_ExecuteEvent;
1092                _cmdButtonManageBooks = new RibbonButton(ribbon1,
(uint)RibbonMarkupCommands.cmdButtonManageBooks);
1093                _cmdButtonManageBooks.ExecuteEvent +=
_cmdButtonManageBooks_ExecuteEvent;
1094                _cmdButtonManagePublishers = new RibbonButton(ribbon1,
(uint)RibbonMarkupCommands.cmdButtonManagePublishers);
1095                _cmdButtonManagePublishers.ExecuteEvent +=
_cmdButtonManagePublishers_ExecuteEvent;
1096                _cmdButtonManageUsers = new RibbonButton(ribbon1,
(uint)RibbonMarkupCommands.cmdButtonManageUsers);
1097                _cmdButtonManageUsers.ExecuteEvent +=
_cmdButtonManageUsers_ExecuteEvent;
1098                _cmdButtonCloseTab = new RibbonButton(ribbon1,
(uint)RibbonMarkupCommands.cmdButtonCloseTab);
1099                _cmdButtonCloseTab.Enabled = false;
1100                _cmdButtonCloseTab.ExecuteEvent +=
_cmdButtonCloseTab_ExecuteEvent;
1101                _cmdButtonCoverImage = new RibbonButton(ribbon1,
(uint)RibbonMarkupCommands.cmdButtonCoverImage);
1102                _cmdButtonCoverImage.ExecuteEvent +=
_cmdButtonCoverImage_ExecuteEvent;
1103                _cmdButtonViewCategories = new RibbonToggleButton(ribbon1,
(uint)RibbonMarkupCommands.cmdButtonViewCategories);
1104                _cmdButtonViewCategories.ExecuteEvent +=
_cmdButtonViewCategories_ExecuteEvent;
1105                _cmdButtonViewCategories.BooleanValue = true;
1106                _cmdButtonViewDetails = new RibbonToggleButton(ribbon1,
(uint)RibbonMarkupCommands.cmdButtonViewDetails);
1107                _cmdButtonViewDetails.ExecuteEvent +=
_cmdButtonViewDetails_ExecuteEvent;
1108                _cmdButtonViewDetails.BooleanValue = true;
1109                _cmdButtonChangeBookCategory = new RibbonButton(ribbon1,
(uint)RibbonMarkupCommands.cmdButtonChangeBookCategory);
1110                _cmdButtonChangeBookCategory.ExecuteEvent +=
_cmdButtonChangeBookCategory_ExecuteEvent;
1111                _cmdButtonViewStatusBar = new RibbonToggleButton(ribbon1,
(uint)RibbonMarkupCommands.cmdButtonViewStatusBar);
1112                _cmdButtonViewStatusBar.BooleanValue = true;
1113                _cmdButtonViewStatusBar.ExecuteEvent +=
_cmdButtonViewStatusBar_ExecuteEvent;
1114                _cmdButtonNewEditor = new RibbonButton(ribbon1,
(uint)RibbonMarkupCommands.cmdButtonNewEditor);
1115                _cmdButtonNewEditor.ExecuteEvent +=
_cmdButtonNewEditor_ExecuteEvent;
1116                _cmdButtonChangeEditor = new RibbonButton(ribbon1,
(uint)RibbonMarkupCommands.cmdButtonChangeEditor);
1117                _cmdButtonChangeEditor.ExecuteEvent +=
_cmdButtonChangeEditor_ExecuteEvent;
1118                _cmdButtonDeleteEditor = new RibbonButton(ribbon1,
(uint)RibbonMarkupCommands.cmdButtonDeleteEditor);
```

```
                _cmdButtonDeleteEditor.ExecuteEvent +=
_cmdButtonDeleteEditor_ExecuteEvent;
                _cmdButtonDownloadBooks = new RibbonButton(ribbon1,
(uint)RibbonMarkupCommands.cmdButtonDownloadBooks);
                _cmdButtonDownloadBooks.ExecuteEvent +=
_cmdButtonDownloadBooks_ExecuteEvent;
                _cmdButtonNewUser = new RibbonButton(ribbon1,
(uint)RibbonMarkupCommands.cmdButtonNewUser);
                _cmdButtonNewUser.ExecuteEvent +=
_cmdButtonNewUser_ExecuteEvent;
                _cmdButtonChangeUser = new RibbonButton(ribbon1,
(uint)RibbonMarkupCommands.cmdButtonChangeUser);
                _cmdButtonChangeUser.ExecuteEvent +=
_cmdButtonChangeUser_ExecuteEvent;
                _cmdButtonDeleteUser = new RibbonButton(ribbon1,
(uint)RibbonMarkupCommands.cmdButtonDeleteUser);
                _cmdButtonDeleteUser.ExecuteEvent +=
_cmdButtonDeleteUser_ExecuteEvent;
                _cmdButtonDeleteBooks = new RibbonButton(ribbon1,
(uint)RibbonMarkupCommands.cmdButtonDeleteBooks);
                _cmdButtonDeleteBooks.ExecuteEvent +=
_cmdButtonDeleteBooks_ExecuteEvent;

                _cmdButtonAddRootCategory = new RibbonButton(ribbon1,
(uint)RibbonMarkupCommands.cmdButtonAddRootCategory);
                _cmdButtonAddRootCategory.ExecuteEvent +=
_cmdButtonAddRootCategory_ExecuteEvent;
                _cmdButtonAddChildCategory = new RibbonButton(ribbon1,
(uint)RibbonMarkupCommands.cmdButtonAddChildCategory);
                _cmdButtonAddChildCategory.ExecuteEvent +=
_cmdButtonAddChildCategory_ExecuteEvent;
                _cmdButtonEditCategory = new RibbonButton(ribbon1,
(uint)RibbonMarkupCommands.cmdButtonEditCategory);
                _cmdButtonEditCategory.ExecuteEvent +=
_cmdButtonEditCategory_ExecuteEvent;
                _cmdButtonRemoveCategory = new RibbonButton(ribbon1,
(uint)RibbonMarkupCommands.cmdButtonRemoveCategory);
                _cmdButtonRemoveCategory.ExecuteEvent +=
_cmdButtonRemoveCategory_ExecuteEvent;

                _cmdButtonExit = new RibbonButton(ribbon1,
(uint)RibbonMarkupCommands.cmdButtonExit);
                _cmdButtonExit.ExecuteEvent +=
_cmdButtonExit_ExecuteEvent;

                // Tab Groups
                _cmdTabHome = new RibbonTabGroup(ribbon1,
(uint)RibbonMarkupCommands.cmdTabHome);
                _cmdTabSelectedBook = new RibbonTabGroup(ribbon1,
(uint)RibbonMarkupCommands.cmdTabSelectedBook);
                _cmdTabView = new RibbonTabGroup(ribbon1,
(uint)RibbonMarkupCommands.cmdTabView);
                _cmdTabSelectedEditor = new RibbonTabGroup(ribbon1,
(uint)RibbonMarkupCommands.cmdTabSelectedEditor);
```

```
1148            _cmdTabSelectedBooks = new RibbonTabGroup(ribbon1,
(uint)RibbonMarkupCommands.cmdTabSelectedBooks);
1149            _cmdTabSelectedCategory = new RibbonTabGroup(ribbon1,
(uint)RibbonMarkupCommands.cmdTabSelectedCategory);
1150            _cmdTabSelectedUser = new RibbonTabGroup(ribbon1,
(uint)RibbonMarkupCommands.cmdTabSelectedUser);
1151
1152            // TabGroup Groups
1153            _cmdTabGroupSelectedBookTools = new
RibbonTabGroup(ribbon1,
(uint)RibbonMarkupCommands.cmdTabGroupSelectedBookTools);
1154            _cmdTabGroupSelectedEditorTools = new
RibbonTabGroup(ribbon1,
(uint)RibbonMarkupCommands.cmdTabGroupSelectedEditorTools);
1155            _cmdTabGroupSelectedBooksTools = new
RibbonTabGroup(ribbon1,
(uint)RibbonMarkupCommands.cmdTabGroupSelectedBooksTools);
1156            _cmdTabGroupSelectedCategoriesTools = new
RibbonTabGroup(ribbon1,
(uint)RibbonMarkupCommands.cmdTabGroupSelectedCategoriesTools);
1157            _cmdTabGroupSelectedUsersTools = new
RibbonTabGroup(ribbon1,
(uint)RibbonMarkupCommands.cmdTabGroupSelectedUsersTools);
1158
1159            // Group Groups
1160            _cmdGroupBooks = new RibbonGroup(ribbon1,
(uint)RibbonMarkupCommands.cmdGroupBooks);
1161            _cmdGroupDownUpLoad = new RibbonGroup(ribbon1,
(uint)RibbonMarkupCommands.cmdGroupDownUpLoad);
1162            _cmdGroupCloseSearch = new RibbonGroup(ribbon1,
(uint)RibbonMarkupCommands.cmdGroupCloseSearch);
1163            _cmdGroupMaintenances = new RibbonGroup(ribbon1,
(uint)RibbonMarkupCommands.cmdGroupMaintenances);
1164            _cmdGroupTabs = new RibbonGroup(ribbon1,
(uint)RibbonMarkupCommands.cmdGroupTabs);
1165            _cmdGroupCover = new RibbonGroup(ribbon1,
(uint)RibbonMarkupCommands.cmdGroupCover);
1166            _cmdGroupVisualization = new RibbonGroup(ribbon1,
(uint)RibbonMarkupCommands.cmdGroupVisualization);
1167            _cmdGroupChangeCategory = new RibbonGroup(ribbon1,
(uint)RibbonMarkupCommands.cmdGroupChangeCategory);
1168            _cmdGroupEditors = new RibbonGroup(ribbon1,
(uint)RibbonMarkupCommands.cmdGroupEditors);
1169            _cmdGroupAddButtons = new RibbonGroup(ribbon1,
(uint)RibbonMarkupCommands.cmdGroupAddButtons);
1170            _cmdGroupUsers = new RibbonGroup(ribbon1,
(uint)RibbonMarkupCommands.cmdGroupUsers);
1171
1172            // Recent Items
1173            _cmdRecentItems = new RibbonRecentItems(ribbon1,
(uint)RibbonMarkupCommands.cmdRecentItems);
1174        }
1175
1176        /// <summary>
```

```
        ///     Métodos de quando um item foi selecionado na
DataGridView 'dgvBooks'
        /// </summary>
        /// <param name="sender"></param>
        /// <param name="e"></param>
        private void uc_Books_ItemSelected(object sender,
SelectedEventArgs e)
        {
            ucBooks uc_Books = ((ucBooks)sender);
            DataGridView dgvBooks = uc_Books.dgvBooks;

            UnSelectContextTab();

            if (e.Selected)
            {
                if (dgvBooks.SelectedRows.Count == 1)
                    _cmdTabGroupSelectedBookTools.ContextAvailable =
ContextAvailability.Active;
                else
                    _cmdTabGroupSelectedBooksTools.ContextAvailable =
ContextAvailability.Active;
            }

            if (dgvBooks != null && dgvBooks.SelectedRows.Count == 1)
            {
                //Obter o ID do livro selecionado
                DataGridViewRow selected_row =
((DataGridViewRow)dgvBooks.SelectedRows[0]);
                int id = Convert.ToInt32(selected_row.Cells[0].Value);

                DS_SQLAccessTableAdapters.BooksTableAdapter ta = new
DS_SQLAccessTableAdapters.BooksTableAdapter();
                DS_SQLAccess.BooksDataTable dt = ta.GetData_Books(id,
null, null, null, null, null, null, null, null, null);

                DS_SQLAccess.BooksRow row_book =
((DS_SQLAccess.BooksRow)dt.Rows[0]);

                if (dgvBooks.SelectedRows.Count == 1)
                {
                    if (row_book != null)
                        _cmdButtonDownloadBook.Enabled =
!row_book.IsfileNull();
                    else
                        _cmdButtonDownloadBook.Enabled = true;
                }
                else
                    _cmdButtonDownloadBook.Enabled = false;
            }

            _cmdButtonChangeBook.Enabled =
(dgvBooks.SelectedRows.Count == 1);
            _cmdButtonUploadBook.Enabled =
(dgvBooks.SelectedRows.Count == 1);
```

```
1220                _cmdButtonCoverImage.Enabled =
(dgvBooks.SelectedRows.Count == 1);
1221
1222                EnableControls();
1223            }
1224
1225            /// <summary>
1226            ///     Métodos de quando um item foi selecionado na
DataGridView 'dgvSearch'
1227            /// </summary>
1228            /// <param name="sender"></param>
1229            /// <param name="e"></param>
1230            private void uc_Search_ItemSelected(object sender,
SelectedEventArgs e)
1231            {
1232                ucBooks uc_Books = ((ucBooks)sender);
1233                DataGridView dgvBooks = uc_Books.dgvBooks;
1234
1235                UnSelectContextTab();
1236
1237                if (e.Selected)
1238                {
1239                    if (dgvBooks.SelectedRows.Count == 1)
1240                        _cmdTabGroupSelectedBookTools.ContextAvailable =
ContextAvailability.Active;
1241                    else
1242                        _cmdTabGroupSelectedBooksTools.ContextAvailable =
ContextAvailability.Active;
1243                }
1244
1245                if (dgvBooks != null && dgvBooks.SelectedRows.Count == 1)
1246                {
1247                    //Obter o ID do livro selecionado
1248                    DataGridViewRow selected_row =
((DataGridViewRow)dgvBooks.SelectedRows[0]);
1249                    int id = Convert.ToInt32(selected_row.Cells[0].Value);
1250
1251                    DS_SQLAccessTableAdapters.BooksTableAdapter ta = new
DS_SQLAccessTableAdapters.BooksTableAdapter();
1252                    DS_SQLAccess.BooksDataTable dt = ta.GetData_Books(id,
null, null, null, null, null, null, null, null, null);
1253
1254                    DS_SQLAccess.BooksRow row_book =
((DS_SQLAccess.BooksRow)dt.Rows[0]);
1255
1256                    if (dgvBooks.SelectedRows.Count == 1)
1257                    {
1258                        if (row_book != null)
1259                            _cmdButtonDownloadBook.Enabled =
!row_book.IsfileNull();
1260                        else
1261                            _cmdButtonDownloadBook.Enabled = true;
1262                    }
1263                    else
```

```
                    _cmdButtonDownloadBook.Enabled = false;
                }

                _cmdButtonChangeBook.Enabled =
(dgvBooks.SelectedRows.Count == 1);
                _cmdButtonUploadBook.Enabled =
(dgvBooks.SelectedRows.Count == 1);
                _cmdButtonCoverImage.Enabled =
(dgvBooks.SelectedRows.Count == 1);

                TabPage tab_search = tabs.SelectedTab;
                tab_search.Text = "Pesquisa" + (uc_Books.filtered ? "
(filtrado)" : string.Empty);

                EnableControls();
            }

            /// <summary>
            ///     Métodos de quando um item foi selecionado na
DataGridView 'dgvEditors'
            /// </summary>
            /// <param name="sender"></param>
            /// <param name="e"></param>
            private void uc_Editors_ItemSelected(object sender,
SelectedEventArgs e)
            {
                ucEditors uc_Editors = ((ucEditors)sender);
                DataGridView dgvEditors =
((DataGridView)Utils.FindControl(tabs, "dgvEditors"));

                UnSelectContextTab();

                if (e.Selected)
                    _cmdTabGroupSelectedEditorTools.ContextAvailable =
ContextAvailability.Active;

                _cmdButtonChangeEditor.Enabled =
(dgvEditors.SelectedRows.Count == 1);
                _cmdButtonDeleteEditor.Enabled =
(dgvEditors.SelectedRows.Count == 1);

                EnableControls();
            }

            /// <summary>
            ///     Métodos de quando um item foi selecionado na
DataGridView 'dgvUsers'
            /// </summary>
            /// <param name="sender"></param>
            /// <param name="e"></param>
            private void uc_Users_ItemSelected(object sender,
SelectedEventArgs e)
            {
                ucUsers uc_Users = ((ucUsers)sender);
```

```
1306                DataGridView dgvUsers =
((DataGridView)Utils.FindControl(tabs, "dgvUsers"));
1307
1308                UnSelectContextTab();
1309
1310                if (e.Selected)
1311                    _cmdTabGroupSelectedUsersTools.ContextAvailable =
ContextAvailability.Active;
1312
1313                int id_selected = -1;
1314                if (dgvUsers.SelectedRows.Count == 1)
1315                    id_selected =
Convert.ToInt32(dgvUsers.SelectedRows[0].Cells[0].Value);
1316
1317                _cmdButtonChangeUser.Enabled =
(dgvUsers.SelectedRows.Count == 1) && (id_selected != 1);
1318                _cmdButtonDeleteUser.Enabled =
(dgvUsers.SelectedRows.Count == 1) && (id_selected != 1);
1319
1320                EnableControls();
1321            }
1322
1323            /// <summary>
1324            ///     Abrir o separador dos livros
1325            /// </summary>
1326            /// <param name="sender"></param>
1327            /// <param name="e"></param>
1328            private void OpenBooksPage(enmTabsMode booksMode)
1329            {
1330                bool searchPageFound = false;
1331                foreach (TabPage tpSearch in tabs.TabPages)
1332                {
1333                    if (tpSearch.Tag != null)
1334                    {
1335                        if (tpSearch.Tag.ToString() ==
Utils.GetEnumDescription(enmTabsMode.Books))
1336                        {
1337                            tabs.SelectedTab = tpSearch;
1338                            searchPageFound = true;
1339                            break;
1340                        }
1341                    }
1342                }
1343
1344                if (!searchPageFound)
1345                {
1346                    TabPage tab_search = new TabPage();
1347                    tabs.TabPages.Add(tab_search);
1348                    tabs.Visible = (tabs.TabPages.Count != 0);
1349                    _cmdButtonCloseTab.Enabled = tabs.Visible;
1350                    picAboutBox.Visible = !tabs.Visible;
1351                    tab_search.ImageIndex = 0;
```

```
                    tab_search.Font = new Font(tab_search.Font.FontFamily,
(float)11.25);
                    tab_search.Text = "Livros";
                    tab_search.Tag =
Utils.GetEnumDescription(enmTabsMode.Books);
                    tab_search.BackColor = Color.FromArgb(0, 252, 252,
252);
                    if (tvCategories.SelectedNode == null)
                        tvCategories.SelectedNode = tvCategories.Nodes[0];
                    ucBooks uc_Books = new ucBooks();
                    if (_cmdButtonViewDetails.BooleanValue == true)
                    {
                        uc_Books.pnlDetails.Visible = true;
                        uc_Books.pnlDetails.Refresh();
                    }
                    uc_Books.category =
((category_info)tvCategories.SelectedNode.Tag).id;
                    uc_Books.Name = "uc_Books";
                    uc_Books.Dock = DockStyle.Fill;
                    uc_Books.ItemSelected += uc_Books_ItemSelected;
                    uc_Books.dgvBooks.MouseClick += dgvBooks_MouseClick;
                    tab_search.Controls.Add(uc_Books);
                    tabs.SelectedTab = tab_search;
                }
                else
                {
                    TabPage tab_search = tabs.SelectedTab;
                    ucBooks uc_Books =
((ucBooks)Utils.FindControl(tabs.SelectedTab,
Utils.GetEnumDescription(enmTabsMode.Books)));

                    if (uc_Books != null)
                    {
                        uc_Books.category =
((category_info)tvCategories.SelectedNode.Tag).id;
                        tab_search.Text = "Livros";

                        if (_cmdButtonViewDetails.BooleanValue == true)
                        {
                            uc_Books.pnlDetails.Visible = true;
                            uc_Books.pnlDetails.Refresh();
                        }
                    }
                }

                EnableControls();
            }

            /// <summary>
            ///     Abrir o separador da pesquisa dos livros
            /// </summary>
            /// <param name="sender"></param>
            /// <param name="e"></param>
```

```
1398          private void OpenBooksSearchPage()
1399          {
1400              bool searchPageFound = false;
1401              foreach (TabPage tpSearch in tabs.TabPages)
1402              {
1403                  if (tpSearch.Tag != null)
1404                  {
1405                      if (tpSearch.Tag.ToString() ==
Utils.GetEnumDescription(enmTabsMode.Search))
1406                      {
1407                          tabs.SelectedTab = tpSearch;
1408                          searchPageFound = true;
1409                          break;
1410                      }
1411                  }
1412              }
1413
1414              if (!searchPageFound)
1415              {
1416                  TabPage tab_search = new TabPage();
1417                  tabs.TabPages.Add(tab_search);
1418                  tabs.Visible = (tabs.TabPages.Count != 0);
1419                  _cmdButtonCloseTab.Enabled = tabs.Visible;
1420                  picAboutBox.Visible = !tabs.Visible;
1421                  tab_search.ImageIndex = 3;
1422                  tab_search.Font = new Font(tab_search.Font.FontFamily,
(float)11.25);
1423                  tab_search.Tag =
Utils.GetEnumDescription(enmTabsMode.Search);
1424                  tab_search.BackColor = Color.FromArgb(0, 252, 252,
252);
1425                  if (tvCategories.SelectedNode == null)
1426                      tvCategories.SelectedNode = tvCategories.Nodes[0];
1427                  ucBooks uc_Books = new ucBooks();
1428                  uc_Books.category = null;
1429                  //tab_search.Text = "Pesquisa";
1430                  tab_search.Text = "Pesquisa" + (uc_Books.filtered ? "
(filtrado)" : string.Empty);
1431                  uc_Books.Name = "uc_Search";
1432                  uc_Books.Dock = DockStyle.Fill;
1433                  uc_Books.ItemSelected += uc_Search_ItemSelected;
1434                  uc_Books.dgvBooks.MouseClick += dgvSearch_MouseClick;
1435                  uc_Books.SearchNow = true;
1436                  tab_search.Controls.Add(uc_Books);
1437                  tabs.SelectedTab = tab_search;
1438              }
1439              else
1440              {
1441                  TabPage tab_search = tabs.SelectedTab;
1442
1443                  string currentMode = tabs.SelectedTab.Tag.ToString();
```

```
                    ucBooks uc_Books =
((ucBooks)Utils.FindControl(tabs.SelectedTab, (currentMode ==
Utils.GetEnumDescription(enmTabsMode.Books) ? "uc_Books" : "uc_Search")));

                    if (uc_Books != null)
                    {
                        uc_Books.category = null;
                        //tab_search.Text = "Pesquisa";
                        tab_search.Text = "Pesquisa" + (uc_Books.filtered
? " (filtrado)" : string.Empty);
                        uc_Books.SearchNow = true;
                    }
                }

                EnableControls();
            }

            /// <summary>
            ///     Abrir o separador das editoras
            /// </summary>
            /// <param name="sender"></param>
            /// <param name="e"></param>
            private void OpenEditorsPage()
            {
                bool editorPageFound = false;
                foreach (TabPage tpEditors in tabs.TabPages)
                {
                    if (tpEditors.Tag != null && tpEditors.Tag.ToString()
== Utils.GetEnumDescription(enmTabsMode.Editors))
                    {
                        tabs.SelectedTab = tpEditors;
                        editorPageFound = true;
                        break;
                    }
                }

                if (!editorPageFound)
                {
                    TabPage tab_editors = new TabPage();
                    tabs.TabPages.Add(tab_editors);
                    tabs.Visible = (tabs.TabPages.Count != 0);
                    _cmdButtonCloseTab.Enabled = tabs.Visible;
                    picAboutBox.Visible = !tabs.Visible;
                    tab_editors.ImageIndex = 1;
                    tab_editors.Font = new
Font(tab_editors.Font.FontFamily, (float)11.25);
                    tab_editors.Text = "Editoras";
                    tab_editors.Tag =
Utils.GetEnumDescription(enmTabsMode.Editors);
                    tab_editors.BackColor = Color.FromArgb(0, 252, 252,
252);
                    ucEditors uc_editors = new ucEditors();
                    uc_editors.Name = "uc_Editors";
```

```
1490                uc_editors.Dock = DockStyle.Fill;
1491                uc_editors.ItemSelected += uc_Editors_ItemSelected;
1492                uc_editors.dgvEditors.MouseClick +=
dgvEditors_MouseClick;
1493                tab_editors.Controls.Add(uc_editors);
1494                tabs.SelectedTab = tab_editors;
1495            }
1496
1497            EnableControls();
1498        }
1499
1500        /// <summary>
1501        ///     Abrir o separador dos utilizadores
1502        /// </summary>
1503        /// <param name="sender"></param>
1504        /// <param name="e"></param>
1505        private void OpenUsersPage()
1506        {
1507            bool userPageFound = false;
1508            foreach (TabPage tpUsers in tabs.TabPages)
1509            {
1510                if (tpUsers.Tag != null && tpUsers.Tag.ToString() ==
Utils.GetEnumDescription(enmTabsMode.Users))
1511                {
1512                    tabs.SelectedTab = tpUsers;
1513                    userPageFound = true;
1514                    break;
1515                }
1516            }
1517
1518            if (!userPageFound)
1519            {
1520                TabPage tab_users = new TabPage();
1521                tabs.TabPages.Add(tab_users);
1522                tabs.Visible = (tabs.TabPages.Count != 0);
1523                _cmdButtonCloseTab.Enabled = tabs.Visible;
1524                picAboutBox.Visible = !tabs.Visible;
1525                tab_users.ImageIndex = 2;
1526                tab_users.Font = new Font(tab_users.Font.FontFamily,
(float)11.25);
1527                tab_users.Text = "Utilizadores";
1528                tab_users.Tag =
Utils.GetEnumDescription(enmTabsMode.Users);
1529                tab_users.BackColor = Color.FromArgb(0, 252, 252,
252);
1530                ucUsers uc_users = new ucUsers();
1531                uc_users.Name = "uc_Users";
1532                uc_users.Dock = DockStyle.Fill;
1533                uc_users.ItemSelected += uc_Users_ItemSelected;
1534                uc_users.dgvUsers.MouseClick += dgvUsers_MouseClick;
1535                tab_users.Controls.Add(uc_users);
1536                tabs.SelectedTab = tab_users;
1537            }
```

```
            EnableControls();
        }

        /// <summary>
        ///     Carregar as categorias da base de dados
        /// </summary>
        /// <param name="sender"></param>
        /// <param name="e"></param>
        private void LoadCategories(object obj, int parent_id)
        {
            Application.DoEvents();

            Thread.Sleep(100);

            dt = ta.GetData_Categories(null, parent_id, null);

            if (dt.Rows.Count == 0)
            {
                if (typeof(TreeNode).IsAssignableFrom(obj.GetType()))
                    ((TreeNode)obj).Collapse();
            }

            if (typeof(TreeView).IsAssignableFrom(obj.GetType()))
                ((TreeView)obj).Nodes.Clear();
            if (typeof(TreeNode).IsAssignableFrom(obj.GetType()))
                ((TreeNode)obj).Nodes.Clear();

            foreach (DS_SQLAccess.CategoriesRow row in dt.Rows)
            {
                TreeNode node = null;
                TreeNode node_load = new TreeNode("A carregar...");

                if (typeof(TreeView).IsAssignableFrom(obj.GetType()))
                    node = ((TreeView)obj).Nodes.Add(row.path);
                if (typeof(TreeNode).IsAssignableFrom(obj.GetType()))
                    node = ((TreeNode)obj).Nodes.Add(row.path);

                node.ImageIndex = 1;
                node.SelectedImageIndex = 1;

                node_load.Tag = "loading";
                node_load.ForeColor = Color.FromArgb(143, 156, 171);
                if (row.id > 1)
                    node.Nodes.Add(node_load);
                else
                {
                    node.ImageIndex = 3;
                    node.SelectedImageIndex = 3;
                }
                node.Tag = new category_info(row.id, row.id_parent,
row.path);
            }
```

```
1591                EnableControls();
1592            }
1593
1594            /// <summary>
1595            ///     Iniciar sessão com o utilizador que está 'logado' no
Windows
1596            /// </summary>
1597            /// <param name="sender"></param>
1598            /// <param name="e"></param>
1599            private void LoginWithCurrentWindowsUser()
1600            {
1601                string loggedUser = Environment.UserDomainName + @"\" +
Environment.UserName;
1602
1603                try
1604                {
1605                    DS_SQLAccessTableAdapters.UsersTableAdapter ta = new
DS_SQLAccessTableAdapters.UsersTableAdapter();
1606                    ta.Connection.ConnectionString =
Utils.GetDBConnectionString();
1607                    DS_SQLAccess.UsersDataTable dt =
ta.GetData_Users(null, loggedUser, null);
1608
1609                    if (dt.Rows.Count == 1)
1610                    {
1611                        DS_SQLAccess.UsersRow userRow =
((DS_SQLAccess.UsersRow)dt.Rows[0]);
1612
1613                        DS_SQLAccessTableAdapters.UsersTableAdapter ta2 =
new DS_SQLAccessTableAdapters.UsersTableAdapter();
1614                        ta2.Connection.ConnectionString =
Utils.GetDBConnectionString();
1615                        DS_SQLAccess.UsersDataTable dt2 =
ta2.GetData_Users(userRow.id, null, null);
1616
1617                        //Atribuir as informações do utilizador às
variáveis globais
1618                        Utils.CurrentUsernameID = userRow.id;
1619                        Utils.CurrentUsername = userRow.username;
1620                        Utils.CurrentUserType = userRow.type;
1621
1622                        tssUserType.Text = Utils.CurrentUsername + " " +
(Utils.CurrentUserType == 1 ? "(Administrador)" : "(Utilizador)");
1623                    }
1624                    else
1625                    {
1626                        //Erro a iniciar sessão / Utilizador ou password
errados
1627                        DialogResult result = MessageBox.Show("Erro ao
iniciar sessão\r\n\r\nO utilizador " + Utils.CurrentUsername + " não tem
permissão para utilizar esta aplicação", "Sem acesso", MessageBoxButtons.OK,
MessageBoxIcon.Exclamation);
1628                        Environment.Exit(0);
1629                    }
```

```
                }
                catch (Exception ex)
                {
                    MessageBox.Show("Erro no inicio de sessão.\r\nErro: "
+ ex.Message, "Erro ao iniciar sessão", MessageBoxButtons.OK,
MessageBoxIcon.Exclamation);
                }
            }

            /// <summary>
            ///     Indisponibilizar todos os separadores de contexto da
Ribbon
            /// </summary>
            /// <param name="sender"></param>
            /// <param name="e"></param>
            private void UnSelectContextTab()
            {
                _cmdTabGroupSelectedBooksTools.ContextAvailable =
ContextAvailability.NotAvailable;
                _cmdTabGroupSelectedBookTools.ContextAvailable =
ContextAvailability.NotAvailable;
                _cmdTabGroupSelectedCategoriesTools.ContextAvailable =
ContextAvailability.NotAvailable;
                _cmdTabGroupSelectedEditorTools.ContextAvailable =
ContextAvailability.NotAvailable;
                _cmdTabGroupSelectedUsersTools.ContextAvailable =
ContextAvailability.NotAvailable;
            }

            /// <summary>
            ///     Quando um separador foi fechado
            /// </summary>
            /// <param name="sender"></param>
            /// <param name="e"></param>
            private void TabClosed(object sender,
TabControlCancelEventArgs e)
            {
                tabs.Visible = (tabs.TabPages.Count != 0);
                _cmdButtonCloseTab.Enabled = tabs.Visible;

                picAboutBox.Visible = !tabs.Visible;
            }

            /// <summary>
            ///     Quando um nó da Treeview é expandido
            /// </summary>
            public void AfterExpand(TreeNode node)
            {
                bool collapsed = false;
                if (node.Nodes.Count == 1 && node.Nodes[0].Tag.ToString()
== "loading")
                {
                    LoadCategories(node, ((category_info)node.Tag).id);
```

```
                if (node.Nodes.Count == 0)
                {
                    node.Expand();
                    node.Collapse();
                    collapsed = true;
                }
            }

            if (collapsed)
            {
                node.ImageIndex = 1;
                node.SelectedImageIndex = 1;
            }
            else
            {
                node.ImageIndex = 2;
                node.SelectedImageIndex = 2;
            }

            EnableControls();
        }
        #endregion

        #region Controls Methods
        /// <summary>
        ///     Quando se mudou de separador selecionado
        /// </summary>
        /// <param name="sender"></param>
        /// <param name="e"></param>
        private void tabs_SelectedIndexChanged(object sender,
EventArgs e)
        {
            string currentMode = string.Empty;
            ucBooks uc_Books = null;
            ucEditors uc_Editors = null;
            ucUsers uc_Users = null;
            DataGridView dgvBooks = null;
            DataGridView dgvEditors = null;
            DataGridView dgvUsers = null;

            UnSelectContextTab();

            TabPage tpage = tabs.SelectedTab;
            if (tpage != null && tpage.Tag != null)
            {
                currentMode = tabs.SelectedTab.Tag.ToString();

                if ((tpage.Tag.ToString() ==
Utils.GetEnumDescription(enmTabsMode.Books)))
                {
                    uc_Books =
((ucBooks)Utils.FindControl(tabs.SelectedTab, (currentMode ==
Utils.GetEnumDescription(enmTabsMode.Books) ? "uc_Books" : "uc_Search")));
```

```
                        bool hasItemSelected =
uc_Books.dgvBooks.SelectedRows.Count > 0;
                        uc_Books.LoadBooks();
                        if (!hasItemSelected)
uc_Books.dgvBooks.ClearSelection();

                        if (_cmdButtonViewDetails.BooleanValue == true)
                        {
                            uc_Books.pnlDetails.Visible = true;
                            uc_Books.pnlDetails.Refresh();
                        }

                        dgvBooks =
((DataGridView)Utils.FindControl(tabs.SelectedTab, "dgvBooks"));
                        if (dgvBooks != null)
                        {
                            if (dgvBooks.SelectedRows.Count > 0)
                            {
                                if (dgvBooks.SelectedRows.Count > 1)

_cmdTabGroupSelectedBooksTools.ContextAvailable = ContextAvailability.Active;
                                else

_cmdTabGroupSelectedBookTools.ContextAvailable = ContextAvailability.Active;
                            }
                        }
                    }
                    else if ((tpage.Tag.ToString() ==
Utils.GetEnumDescription(enmTabsMode.Search)))
                    {
                        uc_Books =
((ucBooks)Utils.FindControl(tabs.SelectedTab, (currentMode ==
Utils.GetEnumDescription(enmTabsMode.Books) ? "uc_Books" : "uc_Search")));
                        uc_Books.LoadBooks();

                        if (_cmdButtonViewDetails.BooleanValue == true)
                        {
                            uc_Books.pnlDetails.Visible = true;
                            uc_Books.pnlDetails.Refresh();
                        }

                        dgvBooks =
((DataGridView)Utils.FindControl(tabs.SelectedTab, "dgvBooks"));
                        if (dgvBooks != null)
                        {
                            if (dgvBooks.SelectedRows.Count > 0)
                            {
                                if (dgvBooks.SelectedRows.Count > 1)

_cmdTabGroupSelectedBooksTools.ContextAvailable = ContextAvailability.Active;
                                else

_cmdTabGroupSelectedBookTools.ContextAvailable = ContextAvailability.Active;
                            }
```

```
1765                        }
1766                    }
1767                    else if ((tpage.Tag.ToString() ==
Utils.GetEnumDescription(enmTabsMode.Editors)))
1768                    {
1769                        uc_Editors =
((ucEditors)Utils.FindControl(tabs.SelectedTab, "uc_Editors"));
1770
1771                        bool hasItemSelected =
uc_Editors.dgvEditors.SelectedRows.Count > 0;
1772                        uc_Editors.LoadEditors();
1773                        if (!hasItemSelected)
uc_Editors.dgvEditors.ClearSelection();
1774
1775                        dgvEditors =
((DataGridView)Utils.FindControl(tabs, "dgvEditors"));
1776                        if (dgvEditors != null)
1777                        {
1778                            if (dgvEditors.SelectedRows.Count >= 1)
1779
_cmdTabGroupSelectedEditorTools.ContextAvailable =
ContextAvailability.Active;
1780                        }
1781                    }
1782                    else if ((tpage.Tag.ToString() ==
Utils.GetEnumDescription(enmTabsMode.Users)))
1783                    {
1784                        uc_Users =
((ucUsers)Utils.FindControl(tabs.SelectedTab, "uc_Users"));
1785
1786                        bool hasItemSelected =
uc_Users.dgvUsers.SelectedRows.Count > 0;
1787                        uc_Users.LoadUsers();
1788                        if (!hasItemSelected)
uc_Users.dgvUsers.ClearSelection();
1789
1790                        dgvUsers = ((DataGridView)Utils.FindControl(tabs,
"dgvUsers"));
1791                        if (dgvUsers != null)
1792                        {
1793                            if (dgvUsers.SelectedRows.Count >= 1)
1794
_cmdTabGroupSelectedUsersTools.ContextAvailable = ContextAvailability.Active;
1795                        }
1796                    }
1797                }
1798
1799                EnableControls();
1800            }
1801
1802            /// <summary>
1803            ///     Quando se redimensionou
1804            /// </summary>
1805            /// <param name="sender"></param>
```

```
        /// <param name="e"></param>
        private void frmMain_Resize(object sender, EventArgs e)
        {
            // Maximum splitter for Panel1
            splitContainerMain.Panel1MinSize = 250;
            if (this.WindowState != FormWindowState.Minimized)
                splitContainerMain.Panel2MinSize =
(splitContainerMain.Width / 2 - 2);

            EnableControls();
        }

        /// <summary>
        ///     Quando se clica com o rato na dgvBooks
        /// </summary>
        /// <param name="sender"></param>
        /// <param name="e"></param>
        private void dgvBooks_MouseClick(object sender, MouseEventArgs
e)
        {
            if (e.Button == MouseButtons.Right)
            {
                DataGridView dgv = ((DataGridView)sender);
                if (dgv != null)
                {
                    System.Drawing.Point p =
dgv.PointToScreen(e.Location);
                    DataGridView.HitTestInfo dgviti = dgv.HitTest(e.X,
e.Y);

                    if (dgviti.ColumnIndex >= 0 && dgviti.RowIndex >=
0)
                    {
                        dgv.ClearSelection();
                        dgv.Rows[dgviti.RowIndex].Selected = true;
                        dgv.Rows[dgviti.RowIndex].Cells[0].Selected =
true;
                    }

                    if (dgv.SelectedRows.Count == 1)
                    {

ribbon1.ShowContextPopup((uint)RibbonMarkupCommands.cmdBooksContextMap, p.X,
p.Y);
                    }
                }
            }

            EnableControls();
        }

        /// <summary>
        ///     Quando se clica com o rato na dgvSearch
        /// </summary>
```

```
1852            /// <param name="sender"></param>
1853            /// <param name="e"></param>
1854            private void dgvSearch_MouseClick(object sender,
MouseEventArgs e)
1855            {
1856                if (e.Button == MouseButtons.Right)
1857                {
1858                    DataGridView dgv = ((DataGridView)sender);
1859                    if (dgv != null)
1860                    {
1861                        System.Drawing.Point p =
dgv.PointToScreen(e.Location);
1862                        DataGridView.HitTestInfo dgviti = dgv.HitTest(e.X,
e.Y);
1863
1864                        if (dgviti.ColumnIndex >= 0 && dgviti.RowIndex >=
0)
1865                        {
1866                            dgv.ClearSelection();
1867                            dgv.Rows[dgviti.RowIndex].Selected = true;
1868                            dgv.Rows[dgviti.RowIndex].Cells[0].Selected =
true;
1869                        }
1870
1871                        if (dgv.SelectedRows.Count == 1)
1872                        {
1873
ribbon1.ShowContextPopup((uint)RibbonMarkupCommands.cmdBooksContextMap, p.X,
p.Y);
1874                        }
1875                    }
1876                }
1877
1878                EnableControls();
1879            }
1880
1881            /// <summary>
1882            ///     Quando se clica com o rato na dgvEditors
1883            /// </summary>
1884            /// <param name="sender"></param>
1885            /// <param name="e"></param>
1886            private void dgvEditors_MouseClick(object sender,
MouseEventArgs e)
1887            {
1888                if (e.Button == MouseButtons.Right)
1889                {
1890                    DataGridView dgv = ((DataGridView)sender);
1891                    if (dgv != null)
1892                    {
1893                        System.Drawing.Point p =
dgv.PointToScreen(e.Location);
1894                        DataGridView.HitTestInfo dgviti = dgv.HitTest(e.X,
e.Y);
1895
```

```
                    if (dgviti.ColumnIndex >= 0 && dgviti.RowIndex >=
0)
                    {
                        dgv.ClearSelection();
                        dgv.Rows[dgviti.RowIndex].Selected = true;
                        dgv.Rows[dgviti.RowIndex].Cells[0].Selected =
true;
                    }

                    if (dgv.SelectedRows.Count == 1)
                    {
ribbon1.ShowContextPopup((uint)RibbonMarkupCommands.cmdEditorsContextMap,
p.X, p.Y);
                    }
                }
            }

            EnableControls();
        }

        /// <summary>
        ///     Quando se clica com o rato na dgvUsers
        /// </summary>
        /// <param name="sender"></param>
        /// <param name="e"></param>
        private void dgvUsers_MouseClick(object sender, MouseEventArgs
e)
        {
            if (e.Button == MouseButtons.Right)
            {
                DataGridView dgv = ((DataGridView)sender);
                if (dgv != null)
                {
                    System.Drawing.Point p =
dgv.PointToScreen(e.Location);
                    DataGridView.HitTestInfo dgviti = dgv.HitTest(e.X,
e.Y);

                    if (dgviti.ColumnIndex >= 0 && dgviti.RowIndex >=
0)
                    {
                        dgv.ClearSelection();
                        dgv.Rows[dgviti.RowIndex].Selected = true;
                        dgv.Rows[dgviti.RowIndex].Cells[0].Selected =
true;
                    }

                    if (dgv.SelectedRows.Count == 1)
                    {
ribbon1.ShowContextPopup((uint)RibbonMarkupCommands.cmdUsersContextMap, p.X,
p.Y);
                    }
```

```
1939                }
1940            }
1941
1942            EnableControls();
1943        }
1944
1945        /// <summary>
1946        ///     Quando se expande um nó da treeview
1947        /// </summary>
1948        /// <param name="sender"></param>
1949        /// <param name="e"></param>
1950        private void tvCategories_AfterExpand(object sender,
TreeViewEventArgs e)
1951        {
1952            AfterExpand(e.Node);
1953        }
1954
1955        /// <summary>
1956        ///     Quando se colapsa um nó da treeview
1957        /// </summary>
1958        /// <param name="sender"></param>
1959        /// <param name="e"></param>
1960        private void tvCategories_AfterCollapse(object sender,
TreeViewEventArgs e)
1961        {
1962            e.Node.ImageIndex = 1;
1963            e.Node.SelectedImageIndex = 1;
1964
1965            EnableControls();
1966        }
1967
1968        /// <summary>
1969        ///     Quando se clica com o rato na treeview
1970        /// </summary>
1971        /// <param name="sender"></param>
1972        /// <param name="e"></param>
1973        private void tvCategories_MouseUp(object sender,
MouseEventArgs e)
1974        {
1975            if (e.Button == MouseButtons.Right)
1976            {
1977                TreeView tv = ((TreeView)sender);
1978                System.Drawing.Point p;
1979
1980                p = tv.PointToScreen(e.Location);
1981
1982                TreeViewHitTestInfo tviti = tv.HitTest(e.X, e.Y);
1983
1984                tv.SelectedNode = tviti.Node;
1985
1986                bool buttonsEnabled = (tviti.Node != null);
1987                //Habilitar ou desabilitar os menus: Add Child
Category, Edit Category e Remove Category
```

```
                _cmdButtonAddChildCategory.Enabled = buttonsEnabled;
                _cmdButtonEditCategory.Enabled = buttonsEnabled;
                _cmdButtonRemoveCategory.Enabled = buttonsEnabled;

ribbon1.ShowContextPopup((uint)RibbonMarkupCommands.cmdCategoriesContextMap,
p.X, p.Y);
            }

            EnableControls();
        }

        /// <summary>
        ///     Quando se seleciona num nó da Treeview
        /// </summary>
        /// <param name="sender"></param>
        /// <param name="e"></param>
        private void tvCategories_AfterSelect(object sender,
TreeViewEventArgs e)
        {
            ucBooks uc_Books = ((ucBooks)Utils.FindControl(tabs,
"uc_Books"));
            if (uc_Books != null && tvCategories.SelectedNode != null
&&
typeof(category_info).IsAssignableFrom(tvCategories.SelectedNode.Tag.GetType(
)))
            {
                uc_Books.category =
((category_info)tvCategories.SelectedNode.Tag).id;
                uc_Books.LoadBooks();
            }

            EnableControls();
        }

        /// <summary>
        ///     Quando se clica numa tecla do teclado 'frmMain'
        /// </summary>
        /// <param name="sender"></param>
        /// <param name="e"></param>
        private void frmMain_KeyDown(object sender, KeyEventArgs e)
        {
            if (e.Control && e.KeyCode == Keys.W)
            {
                e.Handled = e.SuppressKeyPress = true;
                _cmdButtonCloseTab_ExecuteEvent(_cmdButtonCloseTab,
null);
            }

            EnableControls();
        }

        /// <summary>
        ///     Quando se clica numa tecla do teclado 'tvCategories'
```

```
2033          /// </summary>
2034          /// <param name="sender"></param>
2035          /// <param name="e"></param>
2036          private void tvCategories_KeyDown(object sender, KeyEventArgs
e)
2037          {
2038              if (e.KeyCode == Keys.F5)
2039              {
2040                  e.Handled = e.SuppressKeyPress = true;
2041                  string category_fullpath =
tvCategories.SelectedNode.FullPath;
2042                  LoadCategories(tvCategories, -1);
2043                  Utils.ExpandTreeByFullPath(this, tvCategories,
category_fullpath);
2044              }
2045
2046              EnableControls();
2047          }
2048          #endregion
2049      }
2050  }
```

Formulário - frmManBooks.cs

```
1   using System;
2   using System.Windows.Forms;
3
4   namespace CeliSoft.Software
5   {
6       /// <summary>
7       ///     frmManBooks
8       ///     Janela de manutenção (e pesquisa) de livros
9       /// </summary>
10      public partial class frmManBooks : Form
11      {
12          #region Variables
13          private int _idCategory = -1;
14          private string _textCategory = string.Empty;
15          private string _category_fullpath = string.Empty;
16          private bool _SearchNow;
17          #endregion
18
19          #region Properties
20          public int idCategory
21          {
22              get { return _idCategory; }
23              set { _idCategory = value; }
24          }
25          public string textCategory
26          {
27              get { return _textCategory; }
28              set { _textCategory = value; }
29          }
```

```
        public string category_fullpath
        {
            get { return _category_fullpath; }
            set { _category_fullpath = value; }
        }
        public bool SearchNow
        {
            get
            {
                return _SearchNow;
            }
            set
            {
                dtpDocDate.ShowCheckBox = value;
                _SearchNow = value;
            }
        }
        #endregion

        #region Window Constructors
        /// <summary>
        ///     Construtor
        /// </summary>
        public frmManBooks()
        {
            InitializeComponent();

            LoadPublishers();
            LoadDocTypes();
        }

        /// <summary>
        ///     Construtor
        /// </summary>
        /// <param name="cat_info"></param>
        public frmManBooks(category_info cat_info)
        {
            InitializeComponent();

            if (cat_info != null)
            {
                _idCategory = cat_info.id;
                _textCategory = cat_info.path;
            }

            LoadPublishers();
            LoadDocTypes();

            txtCategory.Text = _textCategory;
        }
        #endregion

        #region Window Load
```

```
83          /// <summary>
84          ///     Window Load Method
85          /// </summary>
86          /// <param name="sender"></param>
87          /// <param name="e"></param>
88          private void frmManBooks_Load(object sender, EventArgs e)
89          {
90
91              if (_SearchNow)
92              {
93                  cmbPublishers.Width = cmbDocTypes.Width;
94                  btnNewPublisher.Visible = false;
95              }
96          }
97          #endregion
98
99          #region Methods
100         /// <summary>
101         ///     Carregamento das editoras
102         /// </summary>
103         private void LoadPublishers()
104         {
105             DS_SQLAccessTableAdapters.EditorsTableAdapter ta = new
DS_SQLAccessTableAdapters.EditorsTableAdapter();
106             DS_SQLAccess.EditorsDataTable dt = ta.GetData_Editors(null,
null);
107
108             cmbPublishers.Items.Clear();
109             cmbPublishers.Items.Add(new ComboBoxItem(-1, ""));
110             cmbPublishers.SelectedIndex = 0;
111             foreach (DS_SQLAccess.EditorsRow row in dt.Rows)
112                 cmbPublishers.Items.Add(new ComboBoxItem(row.id,
row.editor));
113         }
114
115         /// <summary>
116         ///     Carregamento dos tipos de documentos
117         /// </summary>
118         private void LoadDocTypes()
119         {
120             int[] doc_types_ids = { 1, 2, 3, 4, 5, 6 };
121             string[] doc_types = { "Livro", "Link", "Jornal", "Revista",
"Imagem", "Manuscrito" };
122
123             cmbDocTypes.Items.Clear();
124             for (int i = 0; i < doc_types_ids.Length; i++)
125                 cmbDocTypes.Items.Add(new ComboBoxItem(doc_types_ids[i],
doc_types[i]));
126             cmbDocTypes.SelectedIndex = 0;
127         }
128         #endregion
129
130         #region Controls Methods
```

```
        /// <summary>
        ///     Botão de criação de uma nova editora
        /// </summary>
        /// <param name="sender"></param>
        /// <param name="e"></param>
        private void btnManage_Click(object sender, EventArgs e)
        {
            frmManEditors f_editors = new frmManEditors();
            f_editors.Text = "Adicionar Editor";
            f_editors.txtEditor.Focus();
            f_editors.btnOK.Text = "Adicionar";
            if (f_editors.ShowDialog() == DialogResult.OK)
            {
                DS_SQLAccessTableAdapters.EditorsTableAdapter ta = new
DS_SQLAccessTableAdapters.EditorsTableAdapter();
                DS_SQLAccess.EditorsDataTable dt =
ta.GetData_Editors(null, null);

                DS_SQLAccess.EditorsRow new_row = dt.NewEditorsRow();
                new_row.editor = f_editors.txtEditor.Text;

                dt.Rows.Add(new_row);

                int ret = ta.Update(dt);

                if (ret == 1)
                {
                    int old_ID = -1;
                    if (cmbPublishers.SelectedItem != null)
                        old_ID =
((ComboBoxItem)cmbPublishers.SelectedItem).ID;

                    LoadPublishers();

                    Utils.SelectComboboxItem(cmbPublishers, new_row.id);
                }
            }
        }

        /// <summary>
        ///     Quando se muda de tipo de documento na combo
        /// </summary>
        /// <param name="sender"></param>
        /// <param name="e"></param>
        private void cmbDocTypes_SelectedIndexChanged(object sender,
EventArgs e)
        {
            int docTypeSelected =
((ComboBoxItem)cmbDocTypes.SelectedItem).ID;

            cmbPublishers.Enabled = (docTypeSelected == 1);
            txtDocSource.Enabled = (docTypeSelected == 2);

            btnNewPublisher.Enabled = cmbPublishers.Enabled;
```

```
180
181             if (docTypeSelected != 1)
182                 cmbPublishers.SelectedIndex = -1;
183             if (docTypeSelected != 2)
184                 txtDocSource.Text = string.Empty;
185         }
186
187         /// <summary>
188         ///     Quando se deseja mudar de categoria
189         /// </summary>
190         /// <param name="sender"></param>
191         /// <param name="e"></param>
192         private void btnSelectCategory_Click(object sender, EventArgs e)
193         {
194             frmSelectCategory f_select = new frmSelectCategory();
195
196             f_select.category_path = category_fullpath;
197             Utils.ExpandTreeByFullPath(((frmMain)this.Owner),
f_select.tvCategories, category_fullpath);
198
199             if (f_select.ShowDialog() == DialogResult.OK)
200             {
201                 idCategory = f_select.selected_id;
202                 textCategory = f_select.selected_category;
203
204                 txtCategory.Text = f_select.selected_category;
205
206                 category_fullpath =
f_select.tvCategories.SelectedNode.FullPath;
207             }
208         }
209
210         /// <summary>
211         ///     Botão de efetivar a operação
212         /// </summary>
213         /// <param name="sender"></param>
214         /// <param name="e"></param>
215         private void btnOK_Click(object sender, EventArgs e)
216         {
217             // Obrigatoriedade de preenchimento de campos
218
219             int docTypeSelected = -1;
220             int editorSelected = -1;
221
222             if (SearchNow == false)
223             {
224                 docTypeSelected =
((ComboBoxItem)cmbDocTypes.SelectedItem).ID;
225
226                 if (cmbPublishers.SelectedItem != null)
227                     editorSelected =
((ComboBoxItem)cmbPublishers.SelectedItem).ID;
228
229                 if (string.IsNullOrEmpty(txtReference.Text) ||
```

```
                    string.IsNullOrEmpty(txtTitle.Text) ||
                    string.IsNullOrEmpty(txtAuthor.Text) ||
                    (docTypeSelected == 1 && cmbPublishers.SelectedIndex
== 0) ||
                    (docTypeSelected == 2 &&
string.IsNullOrEmpty(txtDocSource.Text)) ||
                    idCategory == -1 ||
                    string.IsNullOrEmpty(txtResume.Text))
                {
                    MessageBox.Show("Tem de preencher os campos em
falta.", "Novo Livro", MessageBoxButtons.OK, MessageBoxIcon.Exclamation);
                    DialogResult = DialogResult.None;
                    return;
                }
            }
        }

        /// <summary>
        ///     Foco no campo Resumo
        /// </summary>
        /// <param name="sender"></param>
        /// <param name="e"></param>
        private void txtResume_Enter(object sender, EventArgs e)
        {
            this.AcceptButton = null;
        }

        /// <summary>
        ///     Perda de foco do campo Resumo
        /// </summary>
        /// <param name="sender"></param>
        /// <param name="e"></param>
        private void txtResume_Leave(object sender, EventArgs e)
        {
            this.AcceptButton = btnOK;
        }
        #endregion
    }
}
```

Formulário - frmManCategories.cs

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Windows.Forms;

namespace CeliSoft.Software
{
```

```
12      /// <summary>
13      ///     frmManCategories
14      ///     Janela de manutenção de categorias
15      /// </summary>
16      public partial class frmManCategories : Form
17      {
18          #region Window Constructor
19          public frmManCategories()
20          {
21              InitializeComponent();
22          }
23          #endregion
24
25          #region Controls Methods
26          private void btnOK_Click(object sender, EventArgs e)
27          {
28              // Obrigatoriedade de preenchimento de campos
29              if (string.IsNullOrEmpty(txtCategory.Text))
30              {
31                  MessageBox.Show("O campo 'Categoria' tem de estar
preenchido.", "Nova Categoria", MessageBoxButtons.OK,
MessageBoxIcon.Exclamation);
32                  DialogResult = DialogResult.None;
33                  return;
34              }
35          }
36          #endregion
37      }
38  }
```

Formulário - frmManEditors.cs

```
1   using System;
2   using System.Windows.Forms;
3
4   namespace CeliSoft.Software
5   {
6       /// <summary>
7       ///     frmManEditors
8       ///     Janela de manutenção de editoras
9       /// </summary>
10      public partial class frmManEditors : Form
11      {
12          #region Window Constructor
13          public frmManEditors()
14          {
15              InitializeComponent();
16          }
17          #endregion
18
19          #region Controls Methods
20          private void btnOK_Click(object sender, EventArgs e)
21          {
```

```
            // Obrigatoriedade de preenchimento de campos
            if (string.IsNullOrEmpty(txtEditor.Text))
            {
                MessageBox.Show("O campo 'Editora' tem de estar
preenchido.", "Nova Editora", MessageBoxButtons.OK,
MessageBoxIcon.Exclamation);
                DialogResult = DialogResult.None;
                return;
            }
        }
        #endregion
    }
}
```

Formulário - frmManUsers.cs

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Windows.Forms;

namespace CeliSoft.Software
{
    /// <summary>
    ///     frmManUsers
    ///     Janela de manutenção de utilizadores
    /// </summary>
    public partial class frmManUsers : Form
    {
        #region Window Constructor
        public frmManUsers()
        {
            InitializeComponent();

            LoadUserTypes();
        }
        #endregion

        #region Methods
        public void LoadUserTypes()
        {
            int[] doc_types_ids = { 1, 2 };
            string[] doc_types = { "Administrador", "Utilizador" };

            cmbUserType.Items.Clear();
            for (int i = 0; i < doc_types_ids.Length; i++)
                cmbUserType.Items.Add(new ComboBoxItem(doc_types_ids[i],
doc_types[i]));
            cmbUserType.SelectedIndex = 0;
```

```
        }
        #endregion

        #region Controls Methods
        private void btnOK_Click(object sender, EventArgs e)
        {
            // Obrigatoriedade de preenchimento de campos

            if (txtUsername.Text.IndexOf(@"\") < 0)
            {
                MessageBox.Show("O campo 'Utilizador' tem de ser um
utilizador de domínio com o formato Domínio\\Utilizador.", "Novo Utilizador",
MessageBoxButtons.OK, MessageBoxIcon.Exclamation);
                DialogResult = DialogResult.None;
                return;
            }

            DS_SQLAccessTableAdapters.UsersTableAdapter ta = new
DS_SQLAccessTableAdapters.UsersTableAdapter();
            DS_SQLAccess.UsersDataTable dt = ta.GetData_Users(null,
txtUsername.Text, null);

            if (dt.Rows.Count > 0)
            {
                MessageBox.Show("O utilizador '" + txtUsername.Text + "'
já se encontra registado.", "Novo Utilizador", MessageBoxButtons.OK,
MessageBoxIcon.Exclamation);
                DialogResult = DialogResult.None;
                return;
            }
        }
        #endregion
    }
}
```

Formulário - frmSelectCategory.cs

```
using System.Drawing;
using System.Threading;
using System.Windows.Forms;

namespace CeliSoft.Software
{
    /// <summary>
    ///     frmSelectCategory
    ///     Janela de seleção de categoria
    /// </summary>
    public partial class frmSelectCategory : Form
    {
        #region Variables
        DS_SQLAccessTableAdapters.CategoriesTableAdapter ta = new
DS_SQLAccessTableAdapters.CategoriesTableAdapter();
        DS_SQLAccess.CategoriesDataTable dt = null;

```

```
        private int _selected_id;
        private string _selected_category;
        private string _category_path;
        public frmMain f_main;
        #endregion

        #region Properties
        public int selected_id
        {
            get { return _selected_id; }
            set { _selected_id = value; }
        }
        public string selected_category
        {
            get { return _selected_category; }
            set { _selected_category = value; }
        }
        public string category_path
        {
            get { return _category_path; }
            set { _category_path = value; }
        }
        #endregion

        #region Window Constructors
        public frmSelectCategory()
        {
            InitializeComponent();

            InitializeWindow(null);
        }
        public frmSelectCategory(int selected_id)
        {
            InitializeComponent();

            InitializeWindow(selected_id);
        }
        #endregion

        #region Methods
        public void InitializeWindow(int? selected_id)
        {
            if (selected_id != null)
            {

DS_SQLAccessTableAdapters.Categories_GetCategoriesPathTableAdapter ta = new
DS_SQLAccessTableAdapters.Categories_GetCategoriesPathTableAdapter();
                DS_SQLAccess.Categories_GetCategoriesPathDataTable dt =
ta.GetData_GetCategoriesPath(selected_id);

                DS_SQLAccess.Categories_GetCategoriesPathRow row =
((DS_SQLAccess.Categories_GetCategoriesPathRow)dt.Rows[0]);

```

```
66                  if (row != null)
67                  {
68                      category_path = row.fullpath;
69                  }
70              }
71          }
72          private void LoadCategories(object obj, int parent_id)
73          {
74              Application.DoEvents();
75
76              Thread.Sleep(500);
77
78              dt = ta.GetData_Categories(null, parent_id, null);
79
80              if (dt.Rows.Count == 0)
81              {
82                  if (typeof(TreeNode).IsAssignableFrom(obj.GetType()))
83                      ((TreeNode)obj).Collapse();
84              }
85
86              if (typeof(TreeView).IsAssignableFrom(obj.GetType()))
87                  ((TreeView)obj).Nodes.Clear();
88              if (typeof(TreeNode).IsAssignableFrom(obj.GetType()))
89                  ((TreeNode)obj).Nodes.Clear();
90
91              foreach (DS_SQLAccess.CategoriesRow row in dt.Rows)
92              {
93                  TreeNode node = null;
94                  TreeNode node_load = new TreeNode("A carregar...");
95
96                  if (typeof(TreeView).IsAssignableFrom(obj.GetType()))
97                      node = ((TreeView)obj).Nodes.Add(row.path);
98                  if (typeof(TreeNode).IsAssignableFrom(obj.GetType()))
99                      node = ((TreeNode)obj).Nodes.Add(row.path);
100
101                 node.ImageIndex = 1;
102                 node.SelectedImageIndex = 1;
103
104                 node_load.Tag = "loading";
105                 node_load.ForeColor = Color.FromArgb(143, 156, 171);
106                 if (row.id > 1)
107                     node.Nodes.Add(node_load);
108                 else
109                 {
110                     node.ImageIndex = 3;
111                     node.SelectedImageIndex = 3;
112                 }
113
114                 node.Tag = new category_info(row.id, row.id_parent,
row.path);
115
116                 if (category_path != null)
117                 {
```

```
                    if (category_path.IndexOf(@"\") > 0)
                    {
                        if (row.path == category_path.Substring(0,
category_path.IndexOf(@"\")))
                        {
                            category_path =
category_path.Substring(category_path.IndexOf(@"\") + 1);
                            node.Expand();
                        }
                    }
                    else
                    {
                        if (row.path == category_path)
                            tvCategories.SelectedNode = node;
                    }
                }
            }
        }
        #endregion

        #region Control Methods
        /// <summary>
        ///     Quando se expande um nó da treeview de categorias
        /// </summary>
        /// <param name="sender"></param>
        /// <param name="e"></param>
        private void tvCategories_AfterExpand(object sender,
TreeViewEventArgs e)
        {
            bool collapsed = false;
            if (e.Node.Nodes.Count == 1 &&
e.Node.Nodes[0].Tag.ToString() == "loading")
            {
                LoadCategories(e.Node, ((category_info)e.Node.Tag).id);
                if (e.Node.Nodes.Count == 0)
                {
                    e.Node.Expand();
                    e.Node.Collapse();
                    collapsed = true;
                }
            }

            if (collapsed)
            {
                e.Node.ImageIndex = 1;
                e.Node.SelectedImageIndex = 1;
            }
            else
            {
                e.Node.ImageIndex = 2;
                e.Node.SelectedImageIndex = 2;
            }
        }
```

```
167
168         /// <summary>
169         ///     Quando se colapsa um nó da treeview de categorias
170         /// </summary>
171         /// <param name="sender"></param>
172         /// <param name="e"></param>
173         private void tvCategories_AfterCollapse(object sender,
TreeViewEventArgs e)
174         {
175             e.Node.ImageIndex = 1;
176             e.Node.SelectedImageIndex = 1;
177         }
178
179         /// <summary>
180         ///     Quando se clica com o rato na treeview de categorias
181         /// </summary>
182         /// <param name="sender"></param>
183         /// <param name="e"></param>
184         private void tvCategories_MouseUp(object sender, MouseEventArgs
e)
185         {
186             if (e.Button == MouseButtons.Right)
187             {
188                 TreeView tv = ((TreeView)sender);
189                 System.Drawing.Point p;
190
191                 p = tv.PointToScreen(e.Location);
192
193                 TreeViewHitTestInfo tviti = tv.HitTest(e.X, e.Y);
194
195                 tv.SelectedNode = tviti.Node;
196             }
197         }
198
199         /// <summary>
200         ///     Quando se seleciona um nó da treeview de categorias
201         /// </summary>
202         /// <param name="sender"></param>
203         /// <param name="e"></param>
204         private void tvCategories_AfterSelect(object sender,
TreeViewEventArgs e)
205         {
206             selected_id = -1;
207             selected_category = string.Empty;
208
209             if (tvCategories.SelectedNode != null)
210             {
211                 if
(typeof(category_info).IsAssignableFrom(tvCategories.SelectedNode.Tag.GetType
()))
212                 {
213                     category_info cat_info =
((category_info)tvCategories.SelectedNode.Tag);
```

```
                    selected_id = cat_info.id;
                    selected_category = cat_info.path;
                }
            }
        }

        /// <summary>
        ///     Carregamento da Janela
        /// </summary>
        /// <param name="sender"></param>
        /// <param name="e"></param>
        private void frmSelectCategory_Load(object sender,
System.EventArgs e)
        {
            Application.DoEvents();

            // Load categories
            LoadCategories(tvCategories, -1);

            Utils.ExpandTreeByFullPath(f_main, tvCategories,
category_path);
        }
        #endregion
    }
}
```

Formulário - frmSplash.cs

```
using System;
using System.Drawing;
using System.Windows.Forms;
using System.Drawing.Text;

namespace CeliSoft.Software
{
    /// <summary>
    ///     frmSplash
    ///     Serve para mostrar um primeiro logotipo antes da aplicação em
si iniciar
    /// </summary>
    public partial class frmSplash : Form
    {
        #region Window Constructor
        public frmSplash()
        {
            InitializeComponent();

            // Imagema desenhar
            Image img =
(Image)Properties.Resources.DigitalLibrary_SplashScreen;

            // Iniciar um objeto de desenho
            Graphics grf = Graphics.FromImage(img);
```

```
            grf.SmoothingMode =
System.Drawing.Drawing2D.SmoothingMode.AntiAlias;
            grf.TextRenderingHint = TextRenderingHint.AntiAliasGridFit;
            grf.DrawImage(img, new Point(0, 0));

            // Desenhar o texto da versão
            SizeF versionSize = grf.MeasureString("Versão: " +
Utils.ProductVersion, new Font("Segoe UI", 15, FontStyle.Bold));
            float x = 360;
            float y = 154;
            grf.DrawString("Versão: " + Utils.ProductVersion, new
Font("Segoe UI", 15, FontStyle.Bold), Brushes.White, new PointF(x + 1, y +
1));
            grf.DrawString("Versão: " + Utils.ProductVersion, new
Font("Segoe UI", 15, FontStyle.Bold), Brushes.Black, new PointF(x, y));

            // Inicializar a form que possibilita a transparência da
janela
            alphaFormTransformer1.BackgroundImage = img;
            alphaFormTransformer1.TransformForm(255);

            // Libertação de memória
            img.Dispose();
            img = null;
        }
        #endregion

        #region Controls Methods
        private void tmrClose_Tick(object sender, EventArgs e)
        {
            // Passados os x segundos, fechar esta janela
            this.Close();
        }
        #endregion
    }
}
```

Classe - RibbonContextMenuStrip.cs

```
using RibbonLib;
using System.ComponentModel;
using System.Windows.Forms;

namespace CeliSoft.Software
{
    /// <summary>
    ///     RibbonContextMenuStrip
    ///     Serve para possibilitar usar os menus de contexto da Ribbon
    /// </summary>
    public class RibbonContextMenuStrip : ContextMenuStrip
    {
        #region Variables
        private uint _contextPopupID;
```

```
        private Ribbon _ribbon;
        #endregion

        #region Class Constructor
        public RibbonContextMenuStrip(Ribbon ribbon, uint contextPopupID)
            : base()
        {
            _contextPopupID = contextPopupID;
            _ribbon = ribbon;
        }
        #endregion

        #region Override OnOpening
        protected override void OnOpening(CancelEventArgs e)
        {
            _ribbon.ShowContextPopup(_contextPopupID, Cursor.Position.X,
Cursor.Position.Y);
            e.Cancel = true;
        }
        #endregion
    }
}
```

Controlo - ucBooks.cs

```
using System;
using System.Drawing;
using System.Windows.Forms;

namespace CeliSoft.Software
{
    /// <summary>
    ///     Controlo ucBooks
    ///     Serve para listar e tratar dos registos dos livros
    /// </summary>
    public partial class ucBooks : UserControl
    {
        // A delegate type for hooking up change notifications.
        public delegate void SelectedItemEventHandler(object sender,
SelectedEventArgs e);

        #region Variables
        public event SelectedItemEventHandler ItemSelected;

        public bool loading_datagrid = true;

        private string _reference = null;
        private string _title = null;
        private string _author = null;
        private int? _publisher = null;
        private int? _doc_type = null;
        private string _doc_source = null;
        private DateTime? _doc_date = null;
```

```
28          private int? _category = null;
29          private string _resume = null;
30          private bool _SearchNow;
31
32          private Image _selectedCover;
33          private byte[] _selectedFile;
34          private bool _hasFile;
35          private bool _filtered;
36
37          string ids = string.Empty;
38          #endregion
39
40          #region Properties
41          public string reference
42          {
43              get { return _reference; }
44              set { _reference = value; }
45          }
46          public string title
47          {
48              get { return _title; }
49              set { _title = value; }
50          }
51          public string author
52          {
53              get { return _author; }
54              set { _author = value; }
55          }
56          public int? publisher
57          {
58              get { return _publisher; }
59              set { _publisher = value; }
60          }
61          public int? doc_type
62          {
63              get { return _doc_type; }
64              set { _doc_type = value; }
65          }
66          public string doc_source
67          {
68              get { return _doc_source; }
69              set { _doc_source = value; }
70          }
71          public DateTime? doc_date
72          {
73              get { return _doc_date; }
74              set { _doc_date = value; }
75          }
76          public int? category
77          {
78              get { return _category; }
79              set { _category = value; }
80          }
```

```
        public string resume
        {
            get { return _resume; }
            set { _resume = value; }
        }
        public bool SearchNow
        {
            get
            {
                return _SearchNow;
            }
            set
            {
                tmrOpenSearch.Enabled = value;
                _SearchNow = value;
            }
        }

        private bool GetIfFiltered
        {
            get
            {
                return reference != null ||
                       title != null ||
                       author != null ||
                       publisher != null ||
                       doc_type != null ||
                       doc_source != null ||
                       doc_date != null ||
                       category != null ||
                       resume != null;
            }
        }

        public Image selectedCover
        {
            get { return _selectedCover; }
            set { _selectedCover = value; }
        }
        public byte[] selectedFile
        {
            get { return _selectedFile; }
            set { _selectedFile = value; }
        }
        public bool hasFile
        {
            get { return _hasFile; }
            set { _hasFile = value; }
        }
        public bool filtered
        {
            get { return _filtered; }
            set { _filtered = value; }
```

```
        }
        #endregion

        #region Window Constructor
        /// <summary>
        ///     Constructor
        /// </summary>
        public ucBooks()
        {
            InitializeComponent();
        }
        #endregion

        #region Methods
        /// <summary>
        ///     Carregar Livros
        /// </summary>
        public void LoadBooks()
        {
            DS_SQLAccessTableAdapters.Books_SearchTableAdapter ta = new
DS_SQLAccessTableAdapters.Books_SearchTableAdapter();
            DS_SQLAccess.Books_SearchDataTable dt =
ta.GetData_BooksSearch(reference, title, author, publisher, doc_type,
doc_source, doc_date, category, resume);

            dgvBooks.DataSource = dt;

            dgvBooks.Columns[8].Visible = false;

            if (SearchNow)
            {
                lblNoRowsFound.Text = "Não foram encontrados
registos.\r\nPesquise de novo.";
                lblNoRowsFound.Visible = (dt.Rows.Count == 0);
            }
        }
        #endregion

        #region Controls Methods
        /// <summary>
        ///     Window Load Method
        /// </summary>
        /// <param name="sender"></param>
        /// <param name="e"></param>
        private void ucSearchBooks_Load(object sender, EventArgs e)
        {
            LoadBooks();

            picCover.Image = (Image)Properties.Resources.NoBookCover;

            tmrOpenSearch.Enabled = true;
        }
```

```
        /// <summary>
        ///     Quando se clica com o rato na datagridview dgvBooks
        /// </summary>
        /// <param name="sender"></param>
        /// <param name="e"></param>
        private void dgvBooks_MouseUp(object sender, MouseEventArgs e)
        {
            if (dgvBooks.HitTest(e.X, e.Y).ColumnIndex < 0 &&
                dgvBooks.HitTest(e.X, e.Y).RowIndex < 0)
            {
                dgvBooks.ClearSelection();
            }
        }

        /// <summary>
        ///     Quando se muda de seleção de items na datagridview
dgvBooks
        /// </summary>
        /// <param name="sender"></param>
        /// <param name="e"></param>
        private void dgvBooks_SelectionChanged(object sender, EventArgs
e)
        {
            bool selected = (dgvBooks.SelectedRows.Count >= 1);

            if (ItemSelected != null) // && !(Utils.RemovingTab ==
Utils.GetEnumDescription(frmMain.enmTabsMode.Books)))
                ItemSelected(this, new SelectedEventArgs(selected,
(selected ? dgvBooks.SelectedRows[0].Index : -1)));

            _hasFile = false;
            if (selected)
            {
                //Obter o ID do livro selecionado
                DataGridViewRow selected_row =
((DataGridViewRow)dgvBooks.SelectedRows[0]);
                int id = Convert.ToInt32(selected_row.Cells[0].Value);

                DS_SQLAccessTableAdapters.BooksTableAdapter ta = new
DS_SQLAccessTableAdapters.BooksTableAdapter();
                DS_SQLAccess.BooksDataTable dt = ta.GetData_Books(id,
null, null, null, null, null, null, null, null, null);

                DS_SQLAccess.BooksRow row_book =
((DS_SQLAccess.BooksRow)dt.Rows[0]);

                if (row_book.IscoverNull() == false)
                {
                    Image img =
Utils.ObterImagemDeArrayBytes(row_book.cover);
                    _selectedCover = img;
                }
                else
```

```
                {
                    _selectedCover = Properties.Resources.NoBookCover;
                }

                if (row_book.IsfileNull() == false)
                    _hasFile = row_book.file.Length > 0;

                lblReference.Text = row_book.ref_code;
                lblTitle.Text = row_book.title;
                lblAuthor.Text = row_book.author;
                lblDocType.Text =
Utils.GetDocTypeTextByID(row_book.doc_type);
                lblCategory.Text =
Utils.GetCategoryTextByID(row_book.category);
                lblEditor.Text = (row_book.IseditorNull() ? string.Empty
: Utils.GetEditorTextByID(row_book.editor));
                lblDocSource.Text = (row_book.Isdoc_sourceNull() ?
string.Empty : row_book.doc_source);
                lblDocDate.Text = string.Format("{0:dd-MM-yyyy}",
row_book.doc_date);
                lblResume.Text = row_book.book_resume;
            }
            else
            {
                _selectedCover = Properties.Resources.NoBookCover;

                lblReference.Text = string.Empty;
                lblTitle.Text = string.Empty;
                lblAuthor.Text = string.Empty;
                lblDocType.Text = string.Empty;
                lblCategory.Text = string.Empty;
                lblEditor.Text = string.Empty;
                lblDocSource.Text = string.Empty;
                lblDocDate.Text = string.Empty;
                lblResume.Text = string.Empty;
            }

            picCover.Image = _selectedCover;
            picHasFile.Visible = _hasFile;

            pnlDetails.Refresh();
        }

        /// <summary>
        ///     Quando se ordena por coluna na datagridview dgvBooks
        /// </summary>
        /// <param name="sender"></param>
        /// <param name="e"></param>
        private void dgvBooks_ColumnHeaderMouseClick(object sender,
DataGridViewCellMouseEventArgs e)
        {
            dgvBooks.ClearSelection();
            foreach (DataGridViewRow row in dgvBooks.Rows)
```

```
            {
                if (ids == (row.Cells[0].Value.ToString() +
                           row.Cells[1].Value.ToString() +
                           row.Cells[2].Value.ToString() +
                           row.Cells[3].Value.ToString()))
                {
                    row.Cells[1].Selected = true;
                    row.Selected = true;
                }
            }
        }

        /// <summary>
        ///     Quando se clica com o rato na datagridview dgvBooks
        /// </summary>
        /// <param name="sender"></param>
        /// <param name="e"></param>
        private void dgvBooks_MouseDown(object sender, MouseEventArgs e)
        {
            if (dgvBooks.SelectedRows.Count > 0)
            {
                ids = dgvBooks.SelectedRows[0].Cells[0].Value.ToString()
+
                      dgvBooks.SelectedRows[0].Cells[1].Value.ToString()
+
                      dgvBooks.SelectedRows[0].Cells[2].Value.ToString()
+
dgvBooks.SelectedRows[0].Cells[3].Value.ToString();
            }
        }

        /// <summary>
        ///     Quando se clica com o teclado na datagridview dgvBooks
        /// </summary>
        /// <param name="sender"></param>
        /// <param name="e"></param>
        private void dgvBooks_KeyDown(object sender, KeyEventArgs e)
        {
            switch (e.KeyCode)
            {
                case Keys.F5:
                    LoadBooks();
                    break;
                default:
                    break;
            }
        }

        /// <summary>
        ///     Quando a datagridview dgvBooks terminou o carregamento
        /// </summary>
        /// <param name="sender"></param>
```

```
323         /// <param name="e"></param>
324         private void dgvBooks_DataBindingComplete(object sender,
DataGridViewBindingCompleteEventArgs e)
325         {
326             loading_datagrid = false;
327         }
328 
329         /// <summary>
330         ///     Quando o controlo redimensionou
331         /// </summary>
332         /// <param name="sender"></param>
333         /// <param name="e"></param>
334         private void ucBooks_Resize(object sender, EventArgs e)
335         {
336             pnlDetails.Refresh();
337         }
338 
339         /// <summary>
340         ///     Quando a pesquisa for chamada, abrir a janela de
filtragem
341         /// </summary>
342         /// <param name="sender"></param>
343         /// <param name="e"></param>
344         private void tmrOpenSearch_Tick(object sender, EventArgs e)
345         {
346             tmrOpenSearch.Enabled = false;
347             if (SearchNow)
348             {
349                 frmManBooks f_newBook = new frmManBooks();
350                 f_newBook.Icon = Properties.Resources.Search;
351                 f_newBook.Text = "Procurar Livros";
352                 f_newBook.btnOK.Text = "Procurar";
353                 f_newBook.SearchNow = true;
354                 f_newBook.cmbDocTypes.Items.Insert(0, new ComboBoxItem(-
1, ""));
355                 f_newBook.cmbDocTypes.SelectedIndex = 0;
356                 if (f_newBook.ShowDialog() == DialogResult.OK)
357                 {
358                     if
(string.IsNullOrEmpty(f_newBook.txtReference.Text) == false) reference = "%"
+ f_newBook.txtReference.Text + "%";
359                     if (string.IsNullOrEmpty(f_newBook.txtTitle.Text) ==
false) title = "%" + f_newBook.txtTitle.Text + "%";
360                     if (string.IsNullOrEmpty(f_newBook.txtAuthor.Text)
== false) author = "%" + f_newBook.txtAuthor.Text + "%";
361                     if (f_newBook.cmbPublishers.SelectedItem != null &&
((ComboBoxItem)f_newBook.cmbPublishers.SelectedItem).ID >= 0) publisher =
((ComboBoxItem)f_newBook.cmbPublishers.SelectedItem).ID;
362                     if (f_newBook.cmbDocTypes.SelectedItem != null &&
((ComboBoxItem)f_newBook.cmbDocTypes.SelectedItem).ID >= 0) doc_type =
((ComboBoxItem)f_newBook.cmbDocTypes.SelectedItem).ID;
```

```
                        if
(string.IsNullOrEmpty(f_newBook.txtDocSource.Text) == false) doc_source = "%"
+ f_newBook.txtDocSource.Text + "%";
                        if (f_newBook.dtpDocDate.Checked) doc_date =
f_newBook.dtpDocDate.Value;
                        if (f_newBook.idCategory > -1) category =
f_newBook.idCategory;
                        if (string.IsNullOrEmpty(f_newBook.txtResume.Text)
== false) resume = "%" + f_newBook.txtResume.Text + "%";

                        filtered = GetIfFiltered;

                        LoadBooks();

                        reference = null;
                        title = null;
                        author = null;
                        publisher = null;
                        doc_type = null;
                        doc_source = null;
                        doc_date = null;
                        category = null;
                        resume = null;
                    }
                }
            }
            #endregion
        }
    }
```

Controlo - ucEditors.cs

```
    using System;
    using System.Collections.Generic;
    using System.ComponentModel;
    using System.Drawing;
    using System.Data;
    using System.Linq;
    using System.Text;
    using System.Windows.Forms;

    namespace CeliSoft.Software
    {
        /// <summary>
        ///     Controlo ucEditors
        ///     Serve para listar e tratar dos registos das editoras
        /// </summary>
        public partial class ucEditors : UserControl
        {
            // A delegate type for hooking up change notifications.
            public delegate void SelectedItemEventHandler(object sender,
SelectedEventArgs e);
```

```
21          #region Variables
22          public event SelectedItemEventHandler ItemSelected;
23
24          string ids = string.Empty;
25          #endregion
26
27          #region Window Constructor
28          /// <summary>
29          ///     Constructor
30          /// </summary>
31          public ucEditors()
32          {
33              InitializeComponent();
34          }
35          #endregion
36
37          #region Methods
38          public void LoadEditors()
39          {
40              string editor = (txtEditor.Text.Trim() == string.Empty ?
null : "%" + txtEditor.Text + "%");
41
42              DS_SQLAccessTableAdapters.EditorsSearchTableAdapter ta = new
DS_SQLAccessTableAdapters.EditorsSearchTableAdapter();
43              DS_SQLAccess.EditorsSearchDataTable dt =
ta.GetData_EditorsSearch(editor);
44
45              dgvEditors.DataSource = dt;
46          }
47          #endregion
48
49          #region Controls Methods
50          /// <summary>
51          ///     Filtragem das editoras
52          /// </summary>
53          /// <param name="sender"></param>
54          /// <param name="e"></param>
55          private void btnSearch_Click(object sender, EventArgs e)
56          {
57              LoadEditors();
58              btnClearSearch.Enabled = true;
59          }
60
61          /// <summary>
62          ///     Quando se clica com o rato na datagridview dgvSearch
63          /// </summary>
64          /// <param name="sender"></param>
65          /// <param name="e"></param>
66          private void dgvEditors_MouseUp(object sender, MouseEventArgs e)
67          {
68              if (dgvEditors.HitTest(e.X, e.Y).ColumnIndex < 0 &&
69                  dgvEditors.HitTest(e.X, e.Y).RowIndex < 0)
70              {
```

```
                dgvEditors.ClearSelection();
            }
        }

        /// <summary>
        ///     Quando se muda de seleção de items na datagridview
dgvSearch
        /// </summary>
        /// <param name="sender"></param>
        /// <param name="e"></param>
        private void dgvEditors_SelectionChanged(object sender,
EventArgs e)
        {
            bool selected = (dgvEditors.SelectedRows.Count >= 1);

            if (ItemSelected != null && !(Utils.RemovingTab ==
Utils.GetEnumDescription(frmMain.enmTabsMode.Editors)))
                ItemSelected(this, new SelectedEventArgs(selected,
(selected? dgvEditors.SelectedRows[0].Index : -1)));
        }

        /// <summary>
        ///     Quando se deseja limpar o filtro
        /// </summary>
        /// <param name="sender"></param>
        /// <param name="e"></param>
        private void btnClearSearch_Click(object sender, EventArgs e)
        {
            txtEditor.Text = string.Empty;
            btnSearch_Click(btnSearch, null);
            btnClearSearch.Enabled = false;
        }

        /// <summary>
        ///     Quando se carrega todo o código inicial
        /// </summary>
        /// <param name="sender"></param>
        /// <param name="e"></param>
        private void ucEditors_Load(object sender, EventArgs e)
        {
            LoadEditors();
        }

        /// <summary>
        ///     Quando se ordena por coluna na datagridview dgvSearch
        /// </summary>
        /// <param name="sender"></param>
        /// <param name="e"></param>
        private void dgvEditors_ColumnHeaderMouseClick(object sender,
DataGridViewCellMouseEventArgs e)
        {
            dgvEditors.ClearSelection();
            foreach (DataGridViewRow row in dgvEditors.Rows)
```

```
            {
                if (ids == (row.Cells[0].Value.ToString() +
                           row.Cells[1].Value.ToString()))
                {
                    row.Cells[1].Selected = true;
                    row.Selected = true;
                }
            }
        }

        /// <summary>
        ///     Quando se clica com o rato na datagridview dgvSearch
        /// </summary>
        /// <param name="sender"></param>
        /// <param name="e"></param>
        private void dgvEditors_MouseDown(object sender, MouseEventArgs
e)
        {
            if (dgvEditors.SelectedRows.Count > 0)
            {
                ids =
dgvEditors.SelectedRows[0].Cells[0].Value.ToString() +

dgvEditors.SelectedRows[0].Cells[1].Value.ToString();
            }
        }

        /// <summary>
        ///     Quando se clica com o teclado na datagridview dgvSearch
        /// </summary>
        /// <param name="sender"></param>
        /// <param name="e"></param>
        private void dgvEditors_KeyDown(object sender, KeyEventArgs e)
        {
            switch (e.KeyCode)
            {
                case Keys.F5:
                    LoadEditors();
                    break;
                default:
                    break;
            }
        }
        #endregion
    }
}
```

Controlo - ucUsers.cs

```
using System;
using System.Windows.Forms;

namespace CeliSoft.Software
```

```
5   {
6       /// <summary>
7       ///     Controlo ucUsers
8       ///     Serve para listar e tratar dos registos dos utilizadores
9       /// </summary>
10      public partial class ucUsers : UserControl
11      {
12          // A delegate type for hooking up change notifications.
13          public delegate void SelectedItemEventHandler(object sender,
SelectedEventArgs e);
14
15          #region Variables
16          public event SelectedItemEventHandler ItemSelected;
17
18          string ids = string.Empty;
19          #endregion
20
21          #region Window Constructor
22          /// <summary>
23          ///     Constructor
24          /// </summary>
25          public ucUsers()
26          {
27              InitializeComponent();
28          }
29          #endregion
30
31          #region Methods
32          /// <summary>
33          ///     Carregamento dos utilizadores
34          /// </summary>
35          public void LoadUsers()
36          {
37              string user = (txtUsername.Text.Trim() == string.Empty ?
null : "%" + txtUsername.Text + "%");
38              int? type = null;
39
40              if (cmbUserType.SelectedIndex > 0)
41                  type = ((ComboBoxItem)cmbUserType.SelectedItem).ID;
42
43              DS_SQLAccessTableAdapters.UsersSearchTableAdapter ta = new
DS_SQLAccessTableAdapters.UsersSearchTableAdapter();
44              DS_SQLAccess.UsersSearchDataTable dt =
ta.GetData_UsersSearch(user, type);
45
46              dgvUsers.DataSource = dt;
47          }
48
49          /// <summary>
50          ///     Carregamento dos tipos de utilizadores
51          /// </summary>
52          public void LoadUserTypes()
53          {
```

```
54              int[] doc_types_ids = { 0, 1, 2 };
55              string[] doc_types = { "", "Administrador", "Utilizador" };
56
57              cmbUserType.Items.Clear();
58              for (int i = 0; i < doc_types_ids.Length; i++)
59                  cmbUserType.Items.Add(new ComboBoxItem(doc_types_ids[i],
doc_types[i]));
60              cmbUserType.SelectedIndex = 0;
61          }
62          #endregion
63
64          #region Methods
65          /// <summary>
66          ///     Filtrar dos utilizadores
67          /// </summary>
68          /// <param name="sender"></param>
69          /// <param name="e"></param>
70          private void btnSearch_Click(object sender, EventArgs e)
71          {
72              LoadUsers();
73              btnClearSearch.Enabled = true;
74          }
75
76          /// <summary>
77          ///     Quando se clica com o rato na datagridview dgvUsers
78          /// </summary>
79          /// <param name="sender"></param>
80          /// <param name="e"></param>
81          private void dgvUsers_MouseUp(object sender, MouseEventArgs e)
82          {
83              if (dgvUsers.HitTest(e.X, e.Y).ColumnIndex < 0 &&
84                  dgvUsers.HitTest(e.X, e.Y).RowIndex < 0)
85              {
86                  dgvUsers.ClearSelection();
87              }
88          }
89
90          /// <summary>
91          ///     Quando se muda de seleção de items na datagridview
dgvSearch
92          /// </summary>
93          /// <param name="sender"></param>
94          /// <param name="e"></param>
95          private void dgvUsers_SelectionChanged(object sender, EventArgs
e)
96          {
97              bool selected = (dgvUsers.SelectedRows.Count >= 1);
98
99              if (ItemSelected != null && !(Utils.RemovingTab ==
Utils.GetEnumDescription(frmMain.enmTabsMode.Users)))
100                 ItemSelected(this, new SelectedEventArgs(selected,
(selected ? dgvUsers.SelectedRows[0].Index : -1)));
101         }
```

```
        /// <summary>
        ///     Quando se deseja limpar o filtro
        /// </summary>
        /// <param name="sender"></param>
        /// <param name="e"></param>
        private void btnClearSearch_Click(object sender, EventArgs e)
        {
            txtUsername.Text = string.Empty;
            cmbUserType.SelectedIndex = 0;
            btnSearch_Click(btnSearch, null);
            btnClearSearch.Enabled = false;
        }

        /// <summary>
        ///     Quando se carrega todo o código inicial
        /// </summary>
        /// <param name="sender"></param>
        /// <param name="e"></param>
        private void ucUsers_Load(object sender, EventArgs e)
        {
            LoadUserTypes();
            LoadUsers();
        }

        /// <summary>
        ///     Quando se ordena por coluna na datagridview dgvSearch
        /// </summary>
        /// <param name="sender"></param>
        /// <param name="e"></param>
        private void dgvUsers_ColumnHeaderMouseClick(object sender,
DataGridViewCellMouseEventArgs e)
        {
            dgvUsers.ClearSelection();
            foreach (DataGridViewRow row in dgvUsers.Rows)
            {
                if (ids == (row.Cells[0].Value.ToString() +
                            row.Cells[1].Value.ToString()))
                {
                    row.Cells[1].Selected = true;
                    row.Selected = true;
                }
            }
        }

        /// <summary>
        ///     Quando se clica com o rato na datagridview dgvSearch
        /// </summary>
        /// <param name="sender"></param>
        /// <param name="e"></param>
        private void dgvUsers_MouseDown(object sender, MouseEventArgs e)
        {
            if (dgvUsers.SelectedRows.Count > 0)
```

```
            {
                ids = dgvUsers.SelectedRows[0].Cells[0].Value.ToString()
+
dgvUsers.SelectedRows[0].Cells[1].Value.ToString();
            }
        }

        /// <summary>
        ///     Quando se clica com o teclado na datagridview dgvSearch
        /// </summary>
        /// <param name="sender"></param>
        /// <param name="e"></param>
        private void dgvUsers_KeyDown(object sender, KeyEventArgs e)
        {
            switch (e.KeyCode)
            {
                case Keys.F5:
                    LoadUsers();
                    break;
                default:
                    break;
            }
        }
        #endregion
    }
}
```

Classe - Utils.cs

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Configuration;
using System.Drawing;
using System.IO;
using System.Linq;
using System.Reflection;
using System.Text;
using System.Threading;
using System.Windows.Forms;

namespace CeliSoft.Software
{
    public static class Utils
    {
        #region Variables
        private static int _CurrentUsernameID;
        private static string _CurrentUsername;
        private static int _CurrentUserType;
        private static string _FiltrosSaveAs = "Adobe PDF
(*.pdf)|*.pdf|Imagens (*.png; *.jpg; *.gif; *.bmp)|*.png; *.jpg; *.gif;
*.bmp|Microsoft Office (*.doc; *.docx; *.xls; *.xlsx; *.ppt; *.pptx)|*.doc;
```

```
*.docx; *.xls; *.xlsx; *.ppt; *.pptx|Texto (*.txt)|*.txt|Desenvolvimento
(*.sln; *.cs; *.vb)|*.sln; *.cs; *.vb|Web (*.htm; *.html; *.css; *.vbs;
*.js)|*.htm; *.html; *.css; *.vbs; *.js|Todos os Ficheiros (*.*)|(*.*)";

        public static string RemovingTab = string.Empty;
        #endregion

        #region Properties
        public static int CurrentUsernameID
        {
            get { return _CurrentUsernameID; }
            set { _CurrentUsernameID = value; }
        }
        public static string CurrentUsername
        {
            get { return _CurrentUsername; }
            set { _CurrentUsername = value; }
        }
        public static int CurrentUserType
        {
            get { return _CurrentUserType; }
            set { _CurrentUserType = value; }
        }
        public static string FiltrosSaveAs
        {
            get { return _FiltrosSaveAs; }
            set { _FiltrosSaveAs = value; }
        }
        #endregion

        #region Methods
        /// <summary>
        ///     Selecionar uma combobox por ID
        /// </summary>
        /// <param name="cbox"></param>
        /// <param name="id"></param>
        public static void SelectComboboxItem(ComboBox cbox, int id)
        {
            foreach (ComboBoxItem itm in cbox.Items)
            {
                if (itm.ID == id)
                {
                    cbox.SelectedItem = itm;
                    break;
                }
            }
        }

        /// <summary>
        ///     Selecionar uma combobox por Text
        /// </summary>
        /// <param name="cbox"></param>
        /// <param name="text"></param>
```

```
72          public static void SelectComboboxItem(ComboBox cbox, string
text)
73          {
74              foreach (ComboBoxItem itm in cbox.Items)
75              {
76                  if (itm.Text == text)
77                  {
78                      cbox.SelectedItem = itm;
79                      break;
80                  }
81              }
82          }
83
84          /// <summary>
85          ///     Selecionar uma datagridview by value
86          /// </summary>
87          /// <param name="dgv"></param>
88          /// <param name="value"></param>
89          public static void SelectDataGridViewRow(DataGridView dgv,
object value)
90          {
91              foreach (DataGridViewRow row in dgv.Rows)
92              {
93                  if (row.Cells[0].Value.ToString() == value.ToString())
94                  {
95                      dgv.ClearSelection();
96                      row.Selected = true;
97                      row.Cells[0].Selected = true;
98                  }
99              }
100         }
101
102         /// <summary>
103         ///     Function to get the connection string
104         /// </summary>
105         /// <returns></returns>
106         public static string GetDBConnectionString()
107         {
108             return
ConfigurationManager.ConnectionStrings["CeliSoft.Software.Properties.Settings
.DigitalLibConnectionString"].ConnectionString;
109         }
110
111         /// <summary>
112         ///     Função para obter os bytes de uma imagem
113         /// </summary>
114         /// <param name="caminho"></param>
115         /// <param name="filtro"></param>
116         /// <param name="titulo"></param>
117         /// <param name="filename"></param>
118         /// <returns></returns>
119         internal static byte[] ObterBytesFicheiro(string caminho, string
filtro, string titulo, out string filename)
```

```
        {
            byte[] picbyte = null;
            bool askForImage = (caminho == null);

            filename = null;

            if (askForImage)
            {
                //Abrir a janela de procura da imagem se a imagepath =
null
                OpenFileDialog ofd = new OpenFileDialog();
                ofd.CheckFileExists = true;
                ofd.Filter = filtro;
                ofd.Title = titulo;
                if (ofd.ShowDialog() == DialogResult.OK)
                {
                    //Obter o caminho do ficheiro
                    caminho = ofd.FileName;
                    filename = ofd.FileName;
                }
            }

            //Se a variável imagepath não for nula nem vazia
            if (caminho != null && caminho != string.Empty)
            {
                //Obter o caminho completo da imagem
                caminho = System.IO.Path.GetFullPath(caminho);
                //Retirar alguma configuração de caminho relativo
                caminho = caminho.Replace(@"~\", string.Empty);

                //Abrir a imagem e lê-la para o array de bytes
                FileStream fs;
                fs = new FileStream(caminho, FileMode.Open,
FileAccess.Read);

                picbyte = new byte[fs.Length];

                fs.Read(picbyte, 0, System.Convert.ToInt32(fs.Length));
                fs.Close();
            }
            //Retornar o array de bytes
            return picbyte;
        }

        /// <summary>
        ///     Função para obter uma imagem através de um array de
bytes
        /// </summary>
        /// <param name="array_bytes"></param>
        /// <returns></returns>
        internal static Image ObterImagemDeArrayBytes(byte[]
array_bytes)
        {
```

```
            MemoryStream ms = new MemoryStream(array_bytes);
            Image returnImage = Image.FromStream(ms);
            return returnImage;
        }

        /// <summary>
        ///     Função para gravar uma imagem em disco através de um
array de bytes
        /// </summary>
        /// <param name="ficheiro"></param>
        /// <param name="array_bytes"></param>
        /// <returns></returns>
        internal static bool ConverterBytesEmFicheiro(string ficheiro,
byte[] array_bytes)
        {
            try
            {
                //Abrir ficheiro para escrita
                System.IO.FileStream _FileStream =
                   new System.IO.FileStream(ficheiro,
System.IO.FileMode.Create,
                                            System.IO.FileAccess.Write);
                //Escreve um bloco de bytes usando os dados do array de
bytes
                _FileStream.Write(array_bytes, 0, array_bytes.Length);

                //Fechar o ficheiro
                _FileStream.Close();

                return true;
            }
            catch (Exception _Exception)
            {
                //Erro
                Console.WriteLine("Exception caught in process: {0}",
                                  _Exception.ToString());
            }

            //Ocorreu um erro. Retornar false
            return false;
        }

        /// <summary>
        ///     Obter o registo selecionado
        /// </summary>
        /// <param name="tabPages"></param>
        /// <returns></returns>
        public static DataGridViewRow
GetSelectedRow(TabControl.TabPageCollection tabPages)
        {
            foreach (TabPage tPage in tabPages)
            {
                if (tPage.Tag != null)
```

```
                {
                    if (tPage.Tag.ToString() ==
GetEnumDescription(frmMain.enmTabsMode.Books) ||
                        tPage.Tag.ToString() ==
GetEnumDescription(frmMain.enmTabsMode.Search))
                    {
                        DataGridView dgv =
(DataGridView)FindControl((Control)tPage, "dgvBooks");

                        if (dgv != null)
                        {
                            if (dgv.SelectedRows.Count == 1)
                            {
                                return dgv.SelectedRows[0];
                            }
                        }
                    }
                }
            }
            return null;
        }

        /// <summary>
        ///     Procurar um controlo recursivamente por nome numa Form
        /// </summary>
        /// <param name="form"></param>
        /// <param name="name"></param>
        /// <returns></returns>
        public static Control FindControl(Form form, string name)
        {
            foreach (Control ctrl in form.Controls)
            {
                Control ctrl_out = FindControl(ctrl, name);
                if (ctrl.Name == name)
                {
                    return ctrl;
                }
                if (ctrl_out != null && ctrl_out.Name == name)
                {
                    return ctrl_out;
                }
            }
            return null;
        }

        /// <summary>
        ///     Procurar um controlo recursivamente por nome numa Form
        /// </summary>
        /// <param name="form"></param>
        /// <param name="name"></param>
        /// <returns></returns>
        public static Control FindControl(Control control, string name)
        {
```

```
267             foreach (Control ctrl in control.Controls)
268             {
269                 Control ctrl_out = FindControl(ctrl, name);
270                 if (ctrl.Name == name)
271                 {
272                     return ctrl;
273                 }
274                 if (ctrl_out != null && ctrl_out.Name == name)
275                 {
276                     return ctrl_out;
277                 }
278             }
279             return null;
280         }
281 
282         /// <summary>
283         ///     Obter a descrição em texto do enum selecionado
284         /// </summary>
285         /// <param name="value"></param>
286         /// <returns></returns>
287         public static string GetEnumDescription(Enum value)
288         {
289             FieldInfo fi = value.GetType().GetField(value.ToString());
290 
291             DescriptionAttribute[] attributes =
292 
(DescriptionAttribute[])fi.GetCustomAttributes(typeof(DescriptionAttribute),
false);
293 
294             if (attributes != null && attributes.Length > 0)
295                 return attributes[0].Description;
296             else
297                 return value.ToString();
298         }
299 
300         /// <summary>
301         ///     Obter o texto da categoria por ID
302         /// </summary>
303         /// <param name="idCategory"></param>
304         /// <returns></returns>
305         public static string GetCategoryTextByID(int idCategory)
306         {
307             DS_SQLAccessTableAdapters.CategoriesTableAdapter ta = new
DS_SQLAccessTableAdapters.CategoriesTableAdapter();
308             DS_SQLAccess.CategoriesDataTable dt =
ta.GetData_Categories(idCategory, null, null);
309 
310             if (dt.Rows.Count == 1)
311             {
312                 DS_SQLAccess.CategoriesRow row =
((DS_SQLAccess.CategoriesRow)dt.Rows[0]);
313                 return row.path;
314             }
```

```
            return null;
        }

        /// <summary>
        ///     Obter a class category_info da categoria por ID
        /// </summary>
        /// <param name="idCategory"></param>
        /// <returns></returns>
        public static category_info GetCategoryByID(int idCategory)
        {
            DS_SQLAccessTableAdapters.CategoriesTableAdapter ta = new
DS_SQLAccessTableAdapters.CategoriesTableAdapter();
            DS_SQLAccess.CategoriesDataTable dt =
ta.GetData_Categories(idCategory, null, null);

            if (dt.Rows.Count == 1)
            {
                DS_SQLAccess.CategoriesRow row =
((DS_SQLAccess.CategoriesRow)dt.Rows[0]);
                int? id_book = null;
                return new category_info(row.id, row.id_parent,
row.path);
            }

            return null;
        }

        /// <summary>
        ///     Obter a class category_info da categoria por ID
        /// </summary>
        /// <param name="idCategory"></param>
        /// <returns></returns>
        public static int GetCategoryIDByText(string category_text)
        {
            string[] fullpath = category_text.Split(@"\".ToCharArray());

            DS_SQLAccessTableAdapters.CategoriesTableAdapter ta = new
DS_SQLAccessTableAdapters.CategoriesTableAdapter();
            DS_SQLAccess.CategoriesDataTable dt =
ta.GetData_Categories(null, null, fullpath[fullpath.Length - 1]);

            if (dt.Rows.Count == 1)
            {
                DS_SQLAccess.CategoriesRow row =
((DS_SQLAccess.CategoriesRow)dt.Rows[0]);
                return row.id;
            }

            return -1;
        }

        /// <summary>
        ///     Obter a nome da editora por ID
```

```
362         /// </summary>
363         /// <param name="idEditor"></param>
364         /// <returns></returns>
365         public static string GetEditorTextByID(int idEditor)
366         {
367             DS_SQLAccessTableAdapters.EditorsTableAdapter ta = new
DS_SQLAccessTableAdapters.EditorsTableAdapter();
368             DS_SQLAccess.EditorsDataTable dt =
ta.GetData_Editors(idEditor, null);
369
370             if (dt.Rows.Count == 1)
371             {
372                 DS_SQLAccess.EditorsRow row =
((DS_SQLAccess.EditorsRow)dt.Rows[0]);
373                 return row.editor;
374             }
375
376             return null;
377         }
378
379         /// <summary>
380         ///     Obter o tipo de documento por ID
381         /// </summary>
382         /// <param name="idDocType"></param>
383         /// <returns></returns>
384         public static string GetDocTypeTextByID(int idDocType)
385         {
386             int[] doc_types_ids = { 1, 2, 3, 4, 5, 6 };
387             string[] doc_types = { "Livro", "Link", "Jornal", "Revista",
"Imagem", "Manuscrito" };
388
389             int index = 0;
390             foreach (int doc_type_id in doc_types_ids)
391             {
392                 if (doc_type_id == idDocType)
393                 {
394                     return doc_types[index];
395                 }
396                 index++;
397             }
398             return null;
399         }
400
401         /// <summary>
402         ///     Versão deste produto
403         /// </summary>
404         public static string ProductVersion
405         {
406             get
407             {
408                 string prodVers =
Application.ProductVersion.TrimEnd(".0".ToCharArray());
```

```
                return (prodVers.IndexOf(".") > 0 ? prodVers : prodVers
+ ".0");
            }
        }

        /// <summary>
        ///     Expandir a treeview por um caminho completo
        /// </summary>
        /// <param name="tv"></param>
        /// <param name="category_path"></param>
        public static void ExpandTreeByFullPath(frmMain frm, TreeView
tv, string category_path)
        {
            if (string.IsNullOrEmpty(category_path) == false)
            {
                string[] items =
category_path.Split(@"\".ToCharArray());

                int level = 0;
                foreach (string item in items)
                {
                    if (level == 0)
                        SelectCategoryInNodes(frm, tv, tv.Nodes, item);
                    else
                        SelectCategoryInNodes(frm, tv,
tv.SelectedNode.Nodes, item);
                    level++;
                }
            }
        }

        /// <summary>
        ///     Selecionar a treeview pelo nó selecionado e por um nome
        /// </summary>
        /// <param name="tv"></param>
        /// <param name="nodes"></param>
        /// <param name="name"></param>
        private static void SelectCategoryInNodes(frmMain frm, TreeView
tv, TreeNodeCollection nodes, string name)
        {
            foreach (TreeNode node in nodes)
            {
                if (node.Text == name)
                {
                    tv.SelectedNode = node;
                    tv.SelectedNode.Expand();
                    if (frm != null) frm.AfterExpand(node);
                    break;
                }
            }
        }
        #endregion
    }
```

```
457 }
```

Configuração da Ribbon - RibbonMarkup.xml

```
<Application xmlns="http://schemas.microsoft.com/windows/2009/Ribbon">
  <Application.Commands>

    <Command Name="cmdApplicationMenu" Id="1000" LabelTitle="Ficheiro"
/>

    <!-- FILE GROUP -->
    <Command Id="1001" Name="cmdButtonNewBook"
         LabelTitle="&amp;Novo Livro"
         LabelDescription="Novo Livro"
         TooltipTitle="Novo Livro (Ctrl+N)"
         TooltipDescription="Novo livro"
         Keytip="L">
      <Command.LargeImages>
        <Image>Res/book_pastel_add_32x32.bmp</Image>
      </Command.LargeImages>
      <Command.SmallImages>
        <Image>Res/book_pastel_add_16x16.bmp</Image>
      </Command.SmallImages>
    </Command>
    <Command Id="1002" Name="cmdButtonChangeBook"
         LabelTitle="&amp;Modificar Livro"
         LabelDescription="Modificar Livro"
         TooltipTitle="Modificar Livro"
         TooltipDescription="Modificar Livro na base de dados"
         Keytip="M">
      <Command.LargeImages>
        <Image>Res/book_pastel_edit_32x32.bmp</Image>
      </Command.LargeImages>
      <Command.SmallImages>
        <Image>Res/book_pastel_edit_16x16.bmp</Image>
      </Command.SmallImages>
    </Command>
    <Command Id="1003" Name="cmdButtonDeleteBook"
         LabelTitle="&amp;Remover Livro"
         LabelDescription="Remover Livro"
         TooltipTitle="Remover Livro"
         TooltipDescription="Remover Livro da base de dados"
         Keytip="R">
      <Command.LargeImages>
        <Image>Res/book_pastel_delete_32x32.bmp</Image>
      </Command.LargeImages>
      <Command.SmallImages>
        <Image>Res/book_pastel_delete_16x16.bmp</Image>
      </Command.SmallImages>
    </Command>
    <Command Id="1004" Name="cmdButtonUploadBook"
         LabelTitle="&amp;Carregar Ficheiro"
         LabelDescription="Carregar Ficheiro"
```

```
        TooltipTitle="Carregar Ficheiro"
        TooltipDescription="Carregar Ficheiro para a base de dados"
        Keytip="U">
      <Command.LargeImages>
        <Image>Res/FileDatabase_Upload_32x32.bmp</Image>
      </Command.LargeImages>
      <Command.SmallImages>
        <Image>Res/FileDatabase_Upload_16x16.bmp</Image>
      </Command.SmallImages>
    </Command>
    <Command Id="1005" Name="cmdButtonDownloadBook"
        LabelTitle="Transferir Ficheiro"
        LabelDescription="Transferir Ficheiro"
        TooltipTitle="Transferir Ficheiro"
        TooltipDescription="Transferir Ficheiro para a base de dados"
        Keytip="T">
      <Command.LargeImages>
        <Image>Res/FileDatabase_Download_32x32.bmp</Image>
      </Command.LargeImages>
      <Command.SmallImages>
        <Image>Res/FileDatabase_Download_16x16.bmp</Image>
      </Command.SmallImages>
    </Command>
    <Command Id="1006" Name="cmdButtonSearch"
        LabelTitle="&amp;Procurar"
        LabelDescription="Procurar Livro"
        TooltipTitle="Procurar Livro"
        TooltipDescription="Procurar Livro"
        Keytip="P">
      <Command.LargeImages>
        <Image>Res/Search_32x32.bmp</Image>
      </Command.LargeImages>
      <Command.SmallImages>
        <Image>Res/Search_16x16.bmp</Image>
      </Command.SmallImages>
    </Command>
    <Command Id="1007" Name="cmdButtonManageBooks"
        LabelTitle="Gestão de Liv&amp;ros"
        LabelDescription="Gestão de Livros"
        TooltipTitle="Gestão de Livros"
        TooltipDescription="Gestão de Livros"
        Keytip="R">
      <Command.LargeImages>
        <Image>Res/DigitalLibrary_Books_Pastel_32x32.bmp</Image>
      </Command.LargeImages>
      <Command.SmallImages>
        <Image>Res/DigitalLibrary_Books_Pastel_16x16.bmp</Image>
      </Command.SmallImages>
    </Command>
    <Command Id="1008" Name="cmdButtonManagePublishers"
        LabelTitle="Gestão de E&amp;ditoras"
        LabelDescription="Gestão de Editoras"
        TooltipTitle="Gestão de Editoras"
```

```
        TooltipDescription="Gestão de Editoras"
        Keytip="D">
      <Command.LargeImages>
        <Image>Res/Editors_32x32.bmp</Image>
      </Command.LargeImages>
      <Command.SmallImages>
        <Image>Res/Editors_16x16.bmp</Image>
      </Command.SmallImages>
    </Command>
    <Command Id="1010" Name="cmdButtonManageUsers"
        LabelTitle="Gestão de Ut&amp;amp;ilizadores"
        LabelDescription="Gestão de Utilizadores"
        TooltipTitle="Gestão de Utilizadores"
        TooltipDescription="Gestão de Utilizadores"
        Keytip="I">
      <Command.LargeImages>
        <Image>Res/Users_32x32.bmp</Image>
      </Command.LargeImages>
      <Command.SmallImages>
        <Image>Res/Users_16x16.bmp</Image>
      </Command.SmallImages>
    </Command>
    <Command Id="1011" Name="cmdButtonCloseTab"
        LabelTitle="&amp;amp;Fechar Separador"
        LabelDescription="Fechar Separador"
        TooltipTitle="Fechar Separador"
        TooltipDescription="Fechar Separador"
        Keytip="F">
      <Command.LargeImages>
        <Image>Res/Tabs_Close_32x32.bmp</Image>
      </Command.LargeImages>
      <Command.SmallImages>
        <Image>Res/Tabs_Close_16x16.bmp</Image>
      </Command.SmallImages>
    </Command>
    <Command Id="1012" Name="cmdButtonCoverImage"
        LabelTitle="Capa do &amp;amp;Livro"
        LabelDescription="Capa do Livro"
        TooltipTitle="Capa do Livro"
        TooltipDescription="Capa do Livro"
        Keytip="F">
      <Command.LargeImages>
        <Image>Res/book_white_cover_32x32.bmp</Image>
      </Command.LargeImages>
      <Command.SmallImages>
        <Image>Res/book_white_cover_16x16.bmp</Image>
      </Command.SmallImages>
    </Command>
    <Command Id="1013" Name="cmdButtonViewCategories"
        LabelTitle="Ver &amp;amp;Categorias"
        LabelDescription="Ver Categorias"
        TooltipTitle="Ver Categorias"
        TooltipDescription="Ver Categorias"
```

```
        Keytip="C">
      <Command.LargeImages>
        <Image>Res/app_left_side_32x32.bmp</Image>
      </Command.LargeImages>
      <Command.SmallImages>
        <Image>Res/app_left_side_16x16.bmp</Image>
      </Command.SmallImages>
    </Command>
    <Command Id="1014" Name="cmdButtonViewDetails"
        LabelTitle="Ver &amp;Detalhes"
        LabelDescription="Ver Detalhes"
        TooltipTitle="Ver Detalhes"
        TooltipDescription="Ver Detalhes"
        Keytip="D">
      <Command.LargeImages>
        <Image>Res/app_right_side_details_32x32.bmp</Image>
      </Command.LargeImages>
      <Command.SmallImages>
        <Image>Res/app_right_side_details_16x16.bmp</Image>
      </Command.SmallImages>
    </Command>
    <Command Id="1015" Name="cmdButtonChangeBookCategory"
        LabelTitle="Mud&amp;ar de Categoria"
        LabelDescription="Mudar da categoria do livro selecionado"
        TooltipTitle="Mudar de Categoria"
        TooltipDescription="Mudar a categoria do livro selecionado"
        Keytip="A">
      <Command.LargeImages>
        <Image>Res/Category_Change_32x32.bmp</Image>
      </Command.LargeImages>
      <Command.SmallImages>
        <Image>Res/Category_Change_16x16.bmp</Image>
      </Command.SmallImages>
    </Command>
    <Command Id="1016" Name="cmdButtonViewStatusBar"
        LabelTitle="&amp;Barra de Estado"
        LabelDescription="Barra de Estado"
        TooltipTitle="Barra de Estado"
        TooltipDescription="Barra de Estado"
        Keytip="B">
      <Command.LargeImages>
        <Image>Res/app_statusbar_32x32.bmp</Image>
      </Command.LargeImages>
      <Command.SmallImages>
        <Image>Res/app_statusbar_16x16.bmp</Image>
      </Command.SmallImages>
    </Command>
    <Command Id="1017" Name="cmdButtonAddRootCategory"
        LabelTitle="&amp;Adicionar Categoria"
        LabelDescription="Adicionar Categoria"
        TooltipTitle="Adicionar Categoria"
        TooltipDescription="Adicionar Categoria"
        Keytip="A">
```

```
208       <Command.LargeImages>
209         <Image>Res/AddCategory_32x32.bmp</Image>
210       </Command.LargeImages>
211       <Command.SmallImages>
212         <Image>Res/AddCategory_16x16.bmp</Image>
213       </Command.SmallImages>
214     </Command>
215     <Command Id="1018" Name="cmdButtonAddChildCategory"
216          LabelTitle="Adicionar &amp;Sub Categoria"
217          LabelDescription="Adicionar Sub Categoria"
218          TooltipTitle="Adicionar Sub Categoria"
219          TooltipDescription="Adicionar Sub Categoria"
220          Keytip="S">
221       <Command.LargeImages>
222         <Image>Res/AddSubCategory_32x32.bmp</Image>
223       </Command.LargeImages>
224       <Command.SmallImages>
225         <Image>Res/AddSubCategory_16x16.bmp</Image>
226       </Command.SmallImages>
227     </Command>
228     <Command Id="1019" Name="cmdButtonEditCategory"
229          LabelTitle="&amp;Editar Categoria"
230          LabelDescription="Editar Categoria"
231          TooltipTitle="Editar Categoria"
232          TooltipDescription="Editar Categoria"
233          Keytip="E">
234       <Command.LargeImages>
235         <Image>Res/EditCategory_32x32.bmp</Image>
236       </Command.LargeImages>
237       <Command.SmallImages>
238         <Image>Res/EditCategory_16x16.bmp</Image>
239       </Command.SmallImages>
240     </Command>
241     <Command Id="1020" Name="cmdButtonRemoveCategory"
242          LabelTitle="&amp;Remover Categoria"
243          LabelDescription="Remover Categoria"
244          TooltipTitle="Remover Categoria"
245          TooltipDescription="Remover Categoria"
246          Keytip="R">
247       <Command.LargeImages>
248         <Image>Res/RemoveCategory_32x32.bmp</Image>
249       </Command.LargeImages>
250       <Command.SmallImages>
251         <Image>Res/RemoveCategory_16x16.bmp</Image>
252       </Command.SmallImages>
253     </Command>
254     <Command Id="1021" Name="cmdButtonNewEditor"
255          LabelTitle="&amp;Nova Editora"
256          LabelDescription="Nova Editora"
257          TooltipTitle="Nova Editora"
258          TooltipDescription="Nova Editora"
259          Keytip="E">
260       <Command.LargeImages>
```

```
        <Image>Res/Editors_add_32x32.bmp</Image>
      </Command.LargeImages>
      <Command.SmallImages>
        <Image>Res/Editors_add_16x16.bmp</Image>
      </Command.SmallImages>
    </Command>
    <Command Id="1022" Name="cmdButtonChangeEditor"
         LabelTitle="&amp;Modificar Editora"
         LabelDescription="Modificar Editora"
         TooltipTitle="Modificar Editora"
         TooltipDescription="Modificar Editora"
         Keytip="M">
      <Command.LargeImages>
        <Image>Res/Editors_edit_32x32.bmp</Image>
      </Command.LargeImages>
      <Command.SmallImages>
        <Image>Res/Editors_edit_16x16.bmp</Image>
      </Command.SmallImages>
    </Command>
    <Command Id="1023" Name="cmdButtonDeleteEditor"
         LabelTitle="&amp;Remover Editora"
         LabelDescription="Remover Editora"
         TooltipTitle="Remover Editora"
         TooltipDescription="Remover Editora"
         Keytip="R">
      <Command.LargeImages>
        <Image>Res/Editors_delete_32x32.bmp</Image>
      </Command.LargeImages>
      <Command.SmallImages>
        <Image>Res/Editors_delete_16x16.bmp</Image>
      </Command.SmallImages>
    </Command>
    <Command Id="1024" Name="cmdButtonDownloadBooks"
         LabelTitle="Transferir Ficheiros"
         LabelDescription="Transferir Ficheiros"
         TooltipTitle="Transferir Ficheiros"
         TooltipDescription="Transferir Ficheiros para a base de dados"
         Keytip="T">
      <Command.LargeImages>
        <Image>Res/FileDatabase_Download_32x32.bmp</Image>
      </Command.LargeImages>
      <Command.SmallImages>
        <Image>Res/FileDatabase_Download_16x16.bmp</Image>
      </Command.SmallImages>
    </Command>
    <Command Id="1028" Name="cmdButtonNewUser"
         LabelTitle="Novo &amp;Utilizador"
         LabelDescription="Novo Utilizador"
         TooltipTitle="Novo Utilizador"
         TooltipDescription="Novo Utilizador"
         Keytip="U">
      <Command.LargeImages>
        <Image>Res/People Users_Add_32x32.bmp</Image>
```

```
      </Command.LargeImages>
      <Command.SmallImages>
        <Image>Res/People Users_Add_16x16.bmp</Image>
      </Command.SmallImages>
    </Command>
    <Command Id="1029" Name="cmdButtonChangeUser"
         LabelTitle="&amp;Modificar Utilizador"
         LabelDescription="Modificar Utilizador"
         TooltipTitle="Modificar Utilizador"
         TooltipDescription="Modificar Utilizador"
         Keytip="M">
      <Command.LargeImages>
        <Image>Res/People Users_Edit_32x32.bmp</Image>
      </Command.LargeImages>
      <Command.SmallImages>
        <Image>Res/People Users_Edit_16x16.bmp</Image>
      </Command.SmallImages>
    </Command>
    <Command Id="1030" Name="cmdButtonDeleteUser"
         LabelTitle="&amp;Remover Utilizador"
         LabelDescription="Remover Utilizador"
         TooltipTitle="Remover Utilizador"
         TooltipDescription="Remover Utilizador"
         Keytip="R">
      <Command.LargeImages>
        <Image>Res/People Users_Delete_32x32.bmp</Image>
      </Command.LargeImages>
      <Command.SmallImages>
        <Image>Res/People Users_Delete_16x16.bmp</Image>
      </Command.SmallImages>
    </Command>
    <Command Id="1031" Name="cmdButtonDeleteBooks"
         LabelTitle="&amp;Remover Livros"
         LabelDescription="Remover Livros"
         TooltipTitle="Remover Livros"
         TooltipDescription="Remover Livros da base de dados"
         Keytip="R">
      <Command.LargeImages>
        <Image>Res/book_pastel_delete_32x32.bmp</Image>
      </Command.LargeImages>
      <Command.SmallImages>
        <Image>Res/book_pastel_delete_16x16.bmp</Image>
      </Command.SmallImages>
    </Command>
    <!-- FILE GROUP -->

    <Command Id="1099" Name="cmdButtonExit"
  LabelTitle="&amp;Sair"
  LabelDescription="Sair da aplicação"
  TooltipTitle="Sair"
  TooltipDescription="Sair da aplicação."
         Keytip="S">
      <Command.LargeImages>
```

```
        <Image>Res/Exit_32.bmp</Image>
      </Command.LargeImages>
      <Command.SmallImages>
        <Image>Res/Exit_16.bmp</Image>
      </Command.SmallImages>
    </Command>

    <!-- TABS GROUP -->
    <Command Id="2001" Name="cmdTabHome"
             LabelTitle="Principal"
             Keytip="P">
    </Command>
    <Command Id="2002" Name="cmdTabSelectedBook"
             LabelTitle="Livro Selecionado"
             Keytip="L">
    </Command>
    <Command Id="2003" Name="cmdTabView"
             LabelTitle="Ver"
             Keytip="V">
    </Command>
    <Command Id="2004" Name="cmdTabSelectedEditor"
             LabelTitle="Editora Selecionada"
             Keytip="E">
    </Command>
    <Command Id="2005" Name="cmdTabSelectedBooks"
             LabelTitle="Livros Selecionados"
             Keytip="L">
    </Command>
    <Command Id="2007" Name="cmdTabSelectedUser"
             LabelTitle="Utilizador Selecionado"
             Keytip="U">
    </Command>
    <!-- TABS GROUP -->

    <!-- TABSGROUP CONTEXT GROUP -->
    <Command Id="2010" Name="cmdTabGroupSelectedBookTools"
             LabelTitle="Ferramentas"
             Keytip="F">
    </Command>
    <Command Id="2011" Name="cmdTabGroupSelectedEditorTools"
             LabelTitle="Ferramentas"
             Keytip="F">
    </Command>
    <Command Id="2012" Name="cmdTabGroupSelectedBooksTools"
             LabelTitle="Ferramentas"
             Keytip="F">
    </Command>
    <Command Id="2014" Name="cmdTabGroupSelectedUsersTools"
             LabelTitle="Ferramentas"
             Keytip="F">
    </Command>
    <!-- TABSGROUP CONTEXT GROUP -->
```

```
420     <!-- GROUPS GROUP -->
421     <Command Id="3001" Name="cmdGroupBooks"
422              LabelTitle="Livros"
423              Keytip="L">
424     </Command>
425     <Command Id="3003" Name="cmdGroupDownUpLoad"
426              LabelTitle="Carregar/Transferir"
427              Keytip="T">
428     </Command>
429     <Command Id="3004" Name="cmdGroupSearch"
430              LabelTitle="Procura de Livros"
431              Keytip="P">
432     </Command>
433     <Command Id="3005" Name="cmdGroupCloseSearch"
434              LabelTitle="Fechar Procura"
435              Keytip="F">
436     </Command>
437     <Command Id="3006" Name="cmdGroupMaintenances"
438              LabelTitle="Manutenções"
439              Keytip="M">
440     </Command>
441     <Command Id="3007" Name="cmdGroupTabs"
442              LabelTitle="Separadores"
443              Keytip="S">
444     </Command>
445     <Command Id="3008" Name="cmdGroupCover"
446              LabelTitle="Capa do Ficheiro"
447              Keytip="C">
448     </Command>
449     <Command Id="3009" Name="cmdGroupVisualization"
450              LabelTitle="Visualização"
451              Keytip="V">
452     </Command>
453     <Command Id="3010" Name="cmdGroupChangeCategory"
454              LabelTitle="Categorias"
455              Keytip="C">
456     </Command>
457     <Command Id="3011" Name="cmdGroupEditors"
458              LabelTitle="Editoras"
459              Keytip="E">
460     </Command>
461     <Command Id="3012" Name="cmdGroupAddButtons"
462              LabelTitle="Adicionar"
463              Keytip="A">
464     </Command>
465     <Command Id="3013" Name="cmdGroupUsers"
466              LabelTitle="Utilizadores"
467              Keytip="U">
468     </Command>
469     <!-- GROUPS GROUP -->
470
471     <Command Id="10" Name="cmdQAT"></Command>
472     <Command Id="11" Name="cmdCustomizeQAT"></Command>
```

```
    <Command Id="12" Name="cmdBooksContextMap"></Command>
    <Command Id="13" Name="cmdCategoriesContextMap"></Command>
    <Command Id="14" Name="cmdEditorsContextMap"></Command>
    <Command Id="15" Name="cmdUsersContextMap"></Command>

    <Command Name="cmdRecentItems" Id="9000" LabelTitle=" "
         LabelDescription="Livros adicionados ou editados recentemente"
         TooltipTitle="Livros adicionados ou editados recentemente"
         TooltipDescription="Livros adicionados ou editados
recentemente"
         Keytip="T" />

  </Application.Commands>

  <Application.Views>
    <ContextPopup>
      <ContextPopup.ContextMenus>
        <ContextMenu Name="SelectedBookContextMenu">
          <MenuGroup>
            <Button CommandName="cmdButtonChangeBook" />
            <Button CommandName="cmdButtonDeleteBook" />
          </MenuGroup>
          <MenuGroup>
            <Button CommandName="cmdButtonUploadBook" />
            <Button CommandName="cmdButtonDownloadBook" />
          </MenuGroup>
          <MenuGroup>
            <Button CommandName="cmdButtonCoverImage" />
          </MenuGroup>
          <MenuGroup>
            <ToggleButton CommandName="cmdButtonViewDetails" />
          </MenuGroup>
        </ContextMenu>
        <ContextMenu Name="CategoriesContextMenu">
          <MenuGroup>
            <Button CommandName="cmdButtonAddRootCategory" />
            <Button CommandName="cmdButtonAddChildCategory" />
          </MenuGroup>
          <MenuGroup>
            <Button CommandName="cmdButtonEditCategory" />
          </MenuGroup>
          <MenuGroup>
            <Button CommandName="cmdButtonRemoveCategory" />
          </MenuGroup>
        </ContextMenu>
        <ContextMenu Name="EditorsContextMenu">
          <MenuGroup>
            <Button CommandName="cmdButtonChangeEditor" />
            <Button CommandName="cmdButtonDeleteEditor" />
          </MenuGroup>
        </ContextMenu>
        <ContextMenu Name="UsersContextMenu">
          <MenuGroup>
```

```
525                 <Button CommandName="cmdButtonChangeUser" />
526                 <Button CommandName="cmdButtonDeleteUser" />
527               </MenuGroup>
528             </ContextMenu>
529           </ContextPopup.ContextMenus>
530           <ContextPopup.ContextMaps>
531             <ContextMap CommandName="cmdBooksContextMap"
ContextMenu="SelectedBookContextMenu" />
532             <ContextMap CommandName="cmdCategoriesContextMap"
ContextMenu="CategoriesContextMenu" />
533             <ContextMap CommandName="cmdEditorsContextMap"
ContextMenu="EditorsContextMenu" />
534             <ContextMap CommandName="cmdUsersContextMap"
ContextMenu="UsersContextMenu" />
535           </ContextPopup.ContextMaps>
536         </ContextPopup>
537
538         <Ribbon>
539           <Ribbon.QuickAccessToolbar>
540             <QuickAccessToolbar CommandName="cmdQAT"
CustomizeCommandName="cmdCustomizeQAT">
541               <QuickAccessToolbar.ApplicationDefaults>
542                 <Button CommandName="cmdButtonManageBooks"
ApplicationDefaults.IsChecked="true" />
543                 <Button CommandName="cmdButtonManagePublishers"
ApplicationDefaults.IsChecked="true" />
544                 <Button CommandName="cmdButtonManageUsers"
ApplicationDefaults.IsChecked="true" />
545                 <Button CommandName="cmdButtonNewBook"
ApplicationDefaults.IsChecked="true" />
546               </QuickAccessToolbar.ApplicationDefaults>
547             </QuickAccessToolbar>
548           </Ribbon.QuickAccessToolbar>
549           <Ribbon.ContextualTabs>
550             <TabGroup CommandName="cmdTabGroupSelectedBookTools">
551               <Tab CommandName="cmdTabSelectedBook">
552                 <Group CommandName="cmdGroupBooks"
SizeDefinition="ThreeButtons">
553                   <Button CommandName="cmdButtonNewBook" />
554                   <Button CommandName="cmdButtonChangeBook" />
555                   <Button CommandName="cmdButtonDeleteBook" />
556                 </Group>
557                 <Group CommandName="cmdGroupDownUpLoad"
SizeDefinition="TwoButtons">
558                   <Button CommandName="cmdButtonUploadBook" />
559                   <Button CommandName="cmdButtonDownloadBook" />
560                 </Group>
561                 <Group CommandName="cmdGroupCover"
SizeDefinition="OneButton">
562                   <Button CommandName="cmdButtonCoverImage" />
563                 </Group>
564                 <Group CommandName="cmdGroupChangeCategory"
SizeDefinition="OneButton">
```

```
                <Button CommandName="cmdButtonChangeBookCategory" />
              </Group>
            </Tab>
          </TabGroup>
          <TabGroup CommandName="cmdTabGroupSelectedBooksTools">
            <Tab CommandName="cmdTabSelectedBooks">
              <Group CommandName="cmdGroupBooks"
SizeDefinition="ThreeButtons">
                <Button CommandName="cmdButtonNewBook" />
                <Button CommandName="cmdButtonChangeBook" />
                <Button CommandName="cmdButtonDeleteBooks" />
              </Group>
              <Group CommandName="cmdGroupDownUpLoad"
SizeDefinition="TwoButtons">
                <Button CommandName="cmdButtonUploadBook" />
                <Button CommandName="cmdButtonDownloadBooks" />
              </Group>
              <Group CommandName="cmdGroupCover"
SizeDefinition="OneButton">
                <Button CommandName="cmdButtonCoverImage" />
              </Group>
              <Group CommandName="cmdGroupChangeCategory"
SizeDefinition="OneButton">
                <Button CommandName="cmdButtonChangeBookCategory" />
              </Group>
            </Tab>
          </TabGroup>
          <TabGroup CommandName="cmdTabGroupSelectedEditorTools">
            <Tab CommandName="cmdTabSelectedEditor">
              <Group CommandName="cmdGroupEditors"
SizeDefinition="ThreeButtons">
                <Button CommandName="cmdButtonNewEditor" />
                <Button CommandName="cmdButtonChangeEditor" />
                <Button CommandName="cmdButtonDeleteEditor" />
              </Group>
            </Tab>
          </TabGroup>
          <TabGroup CommandName="cmdTabGroupSelectedUsersTools">
            <Tab CommandName="cmdTabSelectedUser">
              <Group CommandName="cmdGroupUsers"
SizeDefinition="ThreeButtons">
                <Button CommandName="cmdButtonNewUser" />
                <Button CommandName="cmdButtonChangeUser" />
                <Button CommandName="cmdButtonDeleteUser" />
              </Group>
            </Tab>
          </TabGroup>
        </Ribbon.ContextualTabs>
        <Ribbon.Tabs>
          <Tab CommandName="cmdTabHome">
            <Group CommandName="cmdGroupAddButtons"
SizeDefinition="ThreeButtons">
              <Button CommandName="cmdButtonNewBook" />
```

```
            <Button CommandName="cmdButtonNewEditor" />
            <Button CommandName="cmdButtonNewUser" />
          </Group>
          <Group CommandName="cmdGroupTabs" SizeDefinition="OneButton">
            <Button CommandName="cmdButtonCloseTab" />
          </Group>
          <Group CommandName="cmdGroupMaintenances"
SizeDefinition="ThreeButtons">
            <Button CommandName="cmdButtonManageBooks" />
            <Button CommandName="cmdButtonManagePublishers" />
            <Button CommandName="cmdButtonManageUsers" />
          </Group>
          <Group CommandName="cmdGroupSearch"
SizeDefinition="OneButton">
            <Button CommandName="cmdButtonSearch" />
          </Group>
        </Tab>
        <Tab CommandName="cmdTabView">
          <Group CommandName="cmdGroupVisualization"
SizeDefinition="ThreeButtons">
            <ToggleButton CommandName="cmdButtonViewCategories" />
            <ToggleButton CommandName="cmdButtonViewDetails" />
            <ToggleButton CommandName="cmdButtonViewStatusBar" />
          </Group>
        </Tab>
      </Ribbon.Tabs>
      <Ribbon.ApplicationMenu>
        <ApplicationMenu CommandName="cmdApplicationMenu">
          <ApplicationMenu.RecentItems>
            <RecentItems CommandName="cmdRecentItems"
EnablePinning="true" MaxCount="20" />
          </ApplicationMenu.RecentItems>
          <MenuGroup>
            <Button CommandName="cmdButtonNewBook" />
            <Button CommandName="cmdButtonNewEditor" />
            <Button CommandName="cmdButtonNewUser" />
          </MenuGroup>
          <MenuGroup>
            <Button CommandName="cmdButtonCloseTab" />
          </MenuGroup>
          <MenuGroup>
            <Button CommandName="cmdButtonManageBooks" />
            <Button CommandName="cmdButtonManagePublishers" />
            <Button CommandName="cmdButtonManageUsers" />
          </MenuGroup>
          <MenuGroup>
            <Button CommandName="cmdButtonExit" />
          </MenuGroup>
        </ApplicationMenu>
      </Ribbon.ApplicationMenu>
    </Ribbon>
  </Application.Views>
</Application>
```

d) Visual Studio Community 2015 - Acesso ao SQL Server

DS_SQLAccess.xsd

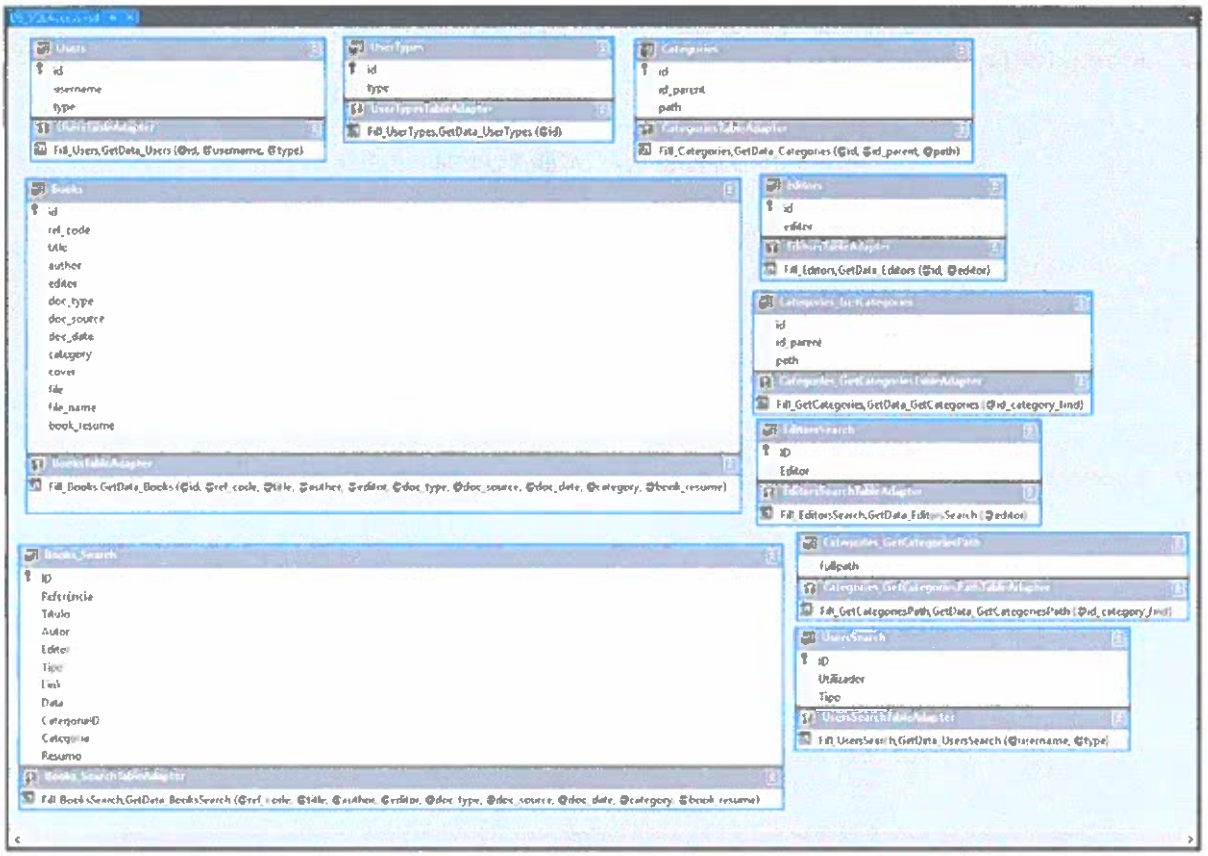

Figura 112 - DataSets de acesso à base de dados por ADO.NET