Licenciatura em Informática

Pós-Laboral

Projeto Global: Biblioteca Digital

Realizado por: Ricardo Pereira

Aluno 1878

Coordenador de Projecto: Prof. Dr. Pedro Brandão

Lisboa

2014/2015

Resumo

As modificações tecnológicas e as recentes conceções para o gerenciamento de recursos de informação têm causado uma alteração no paradigma dos modelos tradicionais de bibliotecas. O conceito de biblioteca digital / virtual apresenta-se como uma alternativa para extender as condições de busca, disponibilidade e recuperação de informações de maneira globalizada, qualitativa, pertinente e racional, aliando o acesso local ao acesso remoto, com base nas redes de telecomunicação disponíveis.

Embora o conceito de biblioteca virtual esteja ainda em construção, deve ser elaborado um planeamento muito cuidadoso, tendo em conta a transição do modelo tradicional de bibliotecas para o modelo de biblioteca virtual. Uma nova abordagem também é exigida para os profissionais bibliotecários e para quem frequenta a biblioteca, visando a um reposicionamento de atitudes e atividades.

Com o advento de novas tecnologias associadas às comunicações em rede, como o ADO.NET, base de dados SQL e virtualização, foram-se alterando vários sectores sociais, no que se refere ao trabalho em ambiente cooperativo, educacional e de acesso a instituições da carácter Público. Neste contexto, o estado da arte apresentado, aborda o potencial destas novas tecnologias e as directrizes de interligação para o projecto da Biblioteca Virtual. Será feita uma análise das tecnologias a utilizar, realçando as vantagens e desvantagens sobre as soluções existentes para a área a desenvolver.

O ADO.NET *(ActiveX Data Objects.NET)*, tecnologia em que a base assenta num conjunto de classes da plataforma .net, cujos componentes foram desenhados para facilitar o acesso, manipulação e tratamento de vários tipos de dados relacionais, documentos XML e dados de aplicações.

O SQL *(Structured Query Language)*, linguagem padrão para gestão e manipulação de dados relacionais através de SGBDS (sistemas de gestão de bases de dados). Permite trabalhar com base de dados: Acces, SQL Server, Oracle, MySql, etc.

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

Palavras chave: Bases de Dados; Virtualização; Sistemas Operativos; Máquinas Virtuais.

Abstract

Technological changes and recent conceptions for information resources management have caused a change in the paradigm of traditional models of libraries. The concept of digital / virtual library presents itself as an alternative to extend the search conditions, availability and retrieval of information in a globalized way, qualitative, relevant and rational, combining local access to remote access, based on telecommunication networks available .

Although the concept of virtual library is still under construction, should be discussed very careful planning, taking into account the transition from the traditional model libraries to the virtual library model. A new approach is also required for librarians and for those attending the library, aimed at a repositioning of attitudes and activities.

With the advent of new technologies associated with network communications, such as ADO.NET, SQL database and virtualization were up changing various social sectors, in relation to work in a cooperative, educational environment and access to institutions Public character. In this context, the state of the art presented, discusses the potential of these new technologies and the interconnection guidelines for the design of the Virtual Library. It will be an analysis of the technologies to be used, highlighting the advantages and disadvantages of existing solutions for the area to develop.

The ADO.NET (ActiveX Data Objects.NET) technology in which the base rests on a set of .NET class platform, whose components are designed for easy access, manipulation and treatment of various types of relational data, XML documents and data applications.

SQL (Structured Query Language) standard language for managing and manipulating relational data using DBMS (database management systems). The program includes database: Acces, SQL Server, Oracle, MySql, etc.

Virtualization, abstraction represented by a computational resource, better known as virtual machine that offers a complete environment, similar to a physical machine, with operating system, applications and network services.

Keywords: Database; Virtualization; Operating Systems; Virtual Machines.

Índice

Índice de ilustrações

Introdução

Para o presente projecto, serão utilizadas tecnologias, que tiram partido do potencial das tecnologias de informação, para a construção de uma aplicação, por forma a interagir com máquinas virtuais no acesso a uma biblioteca virtual, mantida numa base de dados.

A mudança no método da organização do documento para a disponibilidade de informação tem vindo a ser alterada para os diversos tipos de bibliotecas. Diferentes pontos de vista para o gerenciamento de recursos de informação são discutidos, sendo que, destaca-se o conceito de "biblioteca virtual / digital", cuja conceção apresenta-se como uma possível quebra no paradigma de tratamento e disseminação de informações representado pelos recursos, atividades e serviços da "biblioteca tradicional".

Nos dias que correm e com o surgimento e desenvolvimento de novas tecnologias a sobrevivência da biblioteca e o que nos disponibiliza dependem não somente de boas ideias sobre as mudanças apropriadas, mas de cuidadosa atenção sobre como estas mudanças serão implementadas e gerenciadas. Esta sobrevivência far-se-á também tendo em conta certos riscos que são calculados, identificando-se na tecnologia uma oportunidade para melhorar a qualidade dos processos de gestão interna e produtos da biblioteca, que originalmente não foram planeados com um objetivo de eficiência, qualidade, serviço orientado ao cliente e com um objetivo de lucro sobre os investimentos. O fator de risco para a inovação pode ser elevado para as bibliotecas, porém a manutenção do seu status quo, favorecendo a obsolescência, é um risco bastante alto.

Para que se tenha alguma probabilidade de sucesso perante esta conjuntura, o gerente da biblioteca pode adotar metodologias para avaliar e reajustar constantemente o sistema, buscando simplicidade, abrangência e criatividade. Esta postura estratégica implica a percepção, avaliação e possível adoção de perspectivas diferenciadas para a administração de informação que venham ao encontro dos requisitos de qualidade, amplitude, pertinência, racionalização de recursos, custos e tempo envolvidos na coleta, tratamento e disseminação de informação em ambientes cada vez mais dinâmicos.

O modelo tradicional de biblioteca é uma das várias maneiras possíveis de se administrar e gerenciar recursos de informação. Este modelo remonta à história das bibliotecas como guardiãs e depositárias dos registros de conhecimento, o qual se proliferou baseado na idéia de que a exaustividade das coleções permitiria melhor atendimento, pelo

fato de o documento estar à mão quando da demanda do usuário. Neste caso, a busca de informações e documentos fora do ambiente interno das bibliotecas normalmente dependia de catálogos coletivos manuais, nem sempre atualizados e exaustivos, cujos mecanismos de recebimento e envio de documentos eram extremamente morosos quando comparados às atuais condições de intercâmbio atuais. A "explosão de informação" (ou "explosão de documentos"), aliada às novas condições de tratamento, armazenagem e acesso a informações, por meio do uso das tecnologias emergentes deixa de ser apenas clichê e passa a afetar a realidade dos processos tradicionais da maioria das bibliotecas. A definição de diferentes estratégias para o resgate de informações resulta na tomada de decisão, baseada na cuidadosa perceção das condições de tempo, espaço, formato, abrangência, profundidade das demandas de informação por parte dos usuários, da dinâmica dos ambientes internos e externos à biblioteca e das condições de acesso às fontes de informação, no que diz respeito ao seu custo e grau de confiabilidade.

Estado da Arte

Para o projeto em curso serão utilizadas tecnologias que tiram partido do potencial das comunicações em rede, para a construção de uma aplicação, que permite interagir com máquinas virtuais para o acesso a uma biblioteca virtual, mantida numa base de dados.

"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, 2006)

Este estado da arte apresenta uma introdução às tecnologias: *ADO.NET*, SQL, *Cloud Computing* e Virtualização, a ser exploradas e aplicadas ao projeto.

O objetivo deste trabalho será abordar os vários conceitos e a sua importância, expondo as suas vantagens e desvantagens.

ADO.NET

Referencial teórico

O *ADO.NET* é uma tecnologia com base num conjunto de classes, criada pela Microsoft, que segundo Tim Patrick se resume a um modelo de duas grandes classes (uma que contém dados e uma que acede a dados). Como as tecnologias de acesso a dados via web sempre foram essenciais para o rápido desenvolvimento de soluções e cada ambiente de desenvolvimento utiliza dados proprietários como os fornecedores: Oracle, mysql, etc, o *ADO.NET* usa o sistema (*ActiveX Data Objects*[1]))que consiste num conjunto de classes definidas pela *.NET framework* para acesso manipulação e atualiazação de bases de dados armazenadas num servidor remoto. Esta tecnologia permitiu facilitar e poupar tempo aos programadores, separando dados do trabalho dos dados contidos na base dados. Para tal foi desenvolvida uma melhor integração com o XML [2]que permite trabalhar com diferentes tipos de armazenamento de dados (Patrick, October 2010). e que fornece uma ponte entre

[1] É um mecanismo Component Object Model criado pela Microsoft onde os programas o utilizam para a troca de informações com as bases de dados

[2] Serve para gerar linguagens de marcação para necessidades especiais

dados relacionais e hierárquicos. Ao ser associado ao conjunto de dados, ambas as classes podem sincronizar as alterações feitas nos dados contidos nas duas classes.

Este modelo trouxe dois modos de conexão (de acordo com o modelo estabelecido para trabalhar). No modelo desconectado (modelo *Dataset*[3]), um subconjunto de dados pode ser copiado e editado independentemente e mais tarde introduzir as alterações na base dados, idêntico ao trabalho em modo *offline* (Boehm & Mead, 2010). A classe que contém dados é a *"Data Classes"* onde se utiliza o modo desconectado de cesso aos dados. Contém os dados, mas não sabe como obtê-los nem sabe a sua origem, funciona como uma representação em memória de dados relacionais não estando conectada diretamente à base de dados. No modelo conectado *(DataReader*[4]*)* com melhor desempenho, existe uma conexão com a base de dados em que é feita a leitura ou escrita dos dados. A classe que acede aos dados é denominada *".NET data providers*[5]*"*, onde se trabalha de maneira conectada (Boehm & Mead, 2010).

Vantagens do *ADO.NET*

Por ser usado dentro da *.NET Framework*, o *ADO.NET* tira partido de todos os serviços fornecidos pela *Framework*[6], tais como gestão automática de memória, *garbage collection*[7], etc. O que torna o *ADO.NET* fácil de manipular e para quem programa não tem que se preocupar com a gestão da memória usada pelo *ADO.NET*. Para Nuno Ferreira, as vantagens do *ADO.NET* em relação aos seus antecessores, é que existe um suporte para base de dados desconectados, um recurso importante para a criação de aplicações web, aplicações com arquitetura multicamadas etc. È um cenário que apresenta vantagens como: Trabalhar em qualquer altura e ligar-se à base de dados apenas quando necessário; os recursos podem ser usados por outros utilizadores; aumenta a escalabilidade e desempenho das aplicações (Ferreira, ADO.NET).

Desvantagens do *ADO.NET*

Os dados nem sempre estão atualizados porque o *ADO.NET* utiliza dois tipos de objetos para o modelo desconectado no acesso à base de dados: os objetos *Dataset*, que

[3] É uma ampla categoria de objetos usada para ler a partir de uma base de dados
[4] Funciona como uma ponte entre uma base de dados e uma classe de dados desligada
[5] É um provedor de dados usado para se conectar e executar comandos num banco de dados.
[6] É uma abstração que une códigos comuns entre vários projetos de software provendo uma funcionalidade genérica
[7] Processo usado para automatizar a gestão da memória

podem conter um ou mais *Data Table*[8], e os *.NET Data Provider*. Podem ocorrer conflitos de dados que têm que ser resolvidos (Ferreira, ADO.NET, 2004), porque os recursos não são mantidos no servidor durante o processamento dos dados.

Num cenário conectado em que os utilizadores estão permanentemente ligados à bases de dados é necessário haver uma ligação constante ao servidor, o que lhe retira escalabilidade.

SQL

Referencial teórico

A linguagem SQL *(Structured Query Language[9])*, remonta ao início da década de 70, na sequência de um trabalho intitulado *"A Relational Model of Data for Large Shared Data Banks"*, da autoria de Edgar Frank Codd, um investigador da IBM. A ideia era criar um modelo relacional (SGBD) padrão que respondesse racional e eficientemente a inúmeros utilizadores a acederem a grandes volumes de dados, como o SQL é uma linguagem essencialmente declarativa. Significa que o programador necessita apenas de indicar qual o objetivo pretendido para que seja executado pelo SGBDR (Sistema de Gestão de Base de Dados Relacional). (Codd, June 1970)

Por se tratar de uma linguagem padrão, como referiu Júlio Lima, implementa os conceitos definidos no modelo relacional, reduzindo assim as incompatibilidades entre os sistemas e evitando a opção por arquiteturas proprietárias que implicam maiores custos de desenvolvimento e maior esforço financeiro e humano por parte dos intervenientes (Lima, 2012).

Para Fábio Meira a tecnologia aplicada aos métodos de armazenamento de informações tem gerado um impacto cada vez maior no uso de computadores (Meira, 1997). Neste contexto convém referir que ao contrario do processamento tradicional de arquivos, em que a estrutura dos dados está incorporada no programa de acesso e que qualquer alteração na estrutura dos arquivos implica a alteração no código fonte dos programas, na linguagem estruturada é alterada apenas no catálogo, não alterando os programas. Atualmente as bases de dados são coleções de informações relacionadas, para criar um significado dentro de um contexto computacional. Operadas por Sistemas de Gestão de Bases de Dados (SGBDs)

[8] É uma tabela, composta por linhas e colunas que especificam estruturas
[9] Linguagem de Consulta Estruturada; linguagem de pesquisa declarativa padrão para base de dados relacional

como Oracle, MySQL, SQL, etc. As BDs são a base para a maioria das aplicações utilizadas por empresas ou qualquer solução que tenha como base um conjunto de informações que possam ser cruzadas, analisadas, filtradas ou tenham um objetivo específico para o seu armazenamento, como: lista telefónicas, controlo e gestão de uma biblioteca ou sistemas de controlo para departamentos de recursos humanos.

Vantagens do SQL

Tal como James R. Groff e Paul N. Weinberg, no livro *"SQL: The Complete Reference"*, a linguagem SQL, pode ser utilizada por máquinas e sistemas operativos independentes (não Microsoft), proprietários ou não, tais como: computadores pessoais, *mainframes*[10], *Unix*[11], *Aix*[12], etc. Através da linguagem SQL pode-se alterar , expandir ou incluir dinâmicamente estruturas de dados armazenados com bastante flexibilidade, mesmo quando diferentes utilizadores estão a aceder aos seus conteúdos (R. Groff & N. Weinberg, 1999). Ou seja permite, por exemplo, que uma base de dados se adapte a mudanças que ocorram em aplicações on-line sem interrupções.

Desvantagens do SQL

Apesar das vantagens apresentadas, a linguagem SQL, está longe de ser uma linguagem relacional, conforme referiu Christopher J. Date, no livro *"Relational Database: Selected Writings, 1986"*, ao criticar o facto de não existir ortogonalidade nas expressões, funções embutidas, variáveis indicadoras, referência a dados correntes, constante *NULL*[13], conjuntos vazios; Definição formal da linguagem após sua criação; Discordância com as linguagens hospedeiras (geralmente processuais e orientadas para registos e não para conjuntos); Falta de algumas funções; Não dá suporte a alguns aspectos do modelo relacional (join [14]explícito,domínios, e etc.). (J. Date, 1986)

Outro aspeto, que em determinadas aplicações pode ser negativo, é que a linguagem SQL é puramente textual, o que dificulta perceber consultas que envolvam dados que não tenham uma descrição textual adequada. Por exemplo, consultas que envolvam referências a

[10] Computador de grande porte, dedicado ao processamento de um enorme volume de informação
[11] Sistema Operativo multitarefas e multi-utilizador
[12] Sistema operativo UNIX baseado em padrões abertos
[13] Indica valor desconhecido
[14] Junção de duas ou mais tabelas

imagens só podem ser escritas representando uma imagem através de um texto que as identifique, através do nome do arquivo do sistema operativo que as armazena.

Virtualização

Referencial teórico

Segundo Christopher Strachey (cientista da computação), pode dizer-se que a virtualização nasceu em 1959 com a publicação do artigo de *"Time Sharing Processing in large fast computers"*, na Conferência Internacional de processamento de informação, realizada em *Nova York* em 1959 (Strachey, 1959). O artigo focou-se no uso da multiprogramação [15]e o novo conceito para as máquinas de grande porte em que se poderia utilizar melhor os recursos de hardware. Posteriormente com base neste padrão a IBM, introduziu o conceito de multiprocessamento [16]e memória virtual [17]como parte do sistema operativo nos *mainframes*, o que permitiu que vários processadores trabalhassem como um só, dando origem às primeiras formas de virtualização (Veras & Kassick, 2011).

Atualmente podemos definir virtualização como uma metodologia que permite a divisão dos recursos de um computador entre múltiplos ambientes de execução. Como refere Peter Baer Galvin, no artigo *"VMware vSphere Vs. Microsoft Hyper-V, A Technical Analysis. A CTI Strategy White Paper, pag.3, 2009"*, num sistema virtualizado, temos um *software host* em execução na máquina física, chamado *Virtual Machine Monitor (VMM)*, ou *Hypervisor*. Este software é responsável pela criação de ambientes simulados de computação, que são as *VMs (Virtual Machines)*. (Galvin, 2009) Desta forma podemos executar Sistemas Operativos em máquinas virtuais como se estivessem a ser executados num hardware real.

Com a evolução dos sistemas de hardware – aumento da capacidade de processamento, memória, disco, aliado á necessidade crescente de se proceder à execução de tarefas em simultâneo com um custo cada vez menor, fez com que a virtualização aparecesse em maior escala nos últimos anos. Assim, segundo o especialista de vendas de sistemas modulares da IBM, João Almeida, está a ocorrer uma renovação em massa de hardware antigo, com o objetivo de virtualizar a infraestrutura de TI (Almeida, 2011).

---

[15] Técnica para desenvolvimento e execução de mais que um programa de computador em simultâneo, numa mesma máquina

[16] Capacidade de um sistema operativo poder executar simultaneamente dois ou mais processos

[17] Espaço variável e reservado no disco onde o Sistema Operativo armazena dados para auxiliar a memória física

Vantagens da Virtualização

A utilização da virtualização, ao longo dos anos, tem-se revelado uma alternativa interessante em diversos paradigmas da computação, entre eles estão a centralização e consolidação de servidores, as otimizações de hardware e a segurança da informação. Um dos grandes benefícios que traz para uma empresa uma infraestrutura virtual, é que os administradores de TI destinam os recursos dos servidores conforme a estratégia e necessidade do negócio (Tsuguio & Favero de Andrade, 2008). Para Andi Mann , com a consolidação de servidores, em vez das empresas utilizarem vários servidores com os respectivos sistemas operativos, utiliza-se um servidor com máquinas virtuais, alojando vários sistemas operativos independentes com aplicações e serviços, reduzindo-se assim diversos custos administrativos e operacionais (Mann, 2006). O isolamento entre máquinas virtuais e o sistema operativo anfitrião faz com que algumas tarefas arriscadas, sejam seguras. A avaliação de novos sistemas, desenvolvimento de software, testes com vírus e outras ameaças, normalmente significa sujeitar o computador a operações, onde nem sempre é possível reverter os problemas (Goldberg, 1974). É também uma proposta para consolidação de servidores, redução do consumo de energia elétrica em *datacenters*[18], produção de calor, espaço físico e manutenção de equipamentos, com menores custos e menor degradação ambiental (Marshal, 2011).

Desvantagens da Virtualização

Uma das desvantagens prendesse com a segurança. Segundo Thierry Longeau As máquinas virtuais podem ser menos seguras que as máquinas físicas por causa do *host*. Este ponto é interessante, porque se o sistema operativo hospedeiro tiver alguma vulnerabilidade, todas as máquinas virtuais que estão hospedadas nessa máquina física também ficam vulneráveis (Longeau, La Virtualisation des systèmes d´information). Thierry Longeau refere ainda que outra desvantagem se prende com o desempenho, já que atualmente, não existem métodos consolidados para medir o desempenho em ambientes virtualizados. No entanto, a introdução de uma camada extra de software entre o sistema operativo e o hardware, o VMM ou *hypervisor*, gera um custo de processamento superior ao que se teria sem a virtualização. Outro fator importante, tem a ver com a quantidade de máquinas virtuais, já que não se sabe

[18] Instalações seguras que hospedam servidores em rede, armazenamento e recursos de backup

exatamente quantas podem ser executadas por processador, sem que haja prejuízo da qualidade de serviço (Longeau, La Virtualisation des systèmes d'information).

Cloud Computing

Referencial teórico

O termo *Cloud Computing* surgiu em 2006 numa palestra de Eric Schmidt, do Google, ao falar sobre como a empresa geria os *datacenters*. Hoje em dia, o termo *Cloud Computing*, ou Computação na Nuvem, apresenta-se como uma tecnologia que trás profundas mudanças no mundo das tecnologias de informação (Bloor & Jozwiak).

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

O objectivo seguinte será abordar o conceito *Cloud Computing*, a sua importância e a sua aplicação, expondo as vantagens e desvantagens deste tipo de serviços.

Conceito Cloud

Cloud Computing, não é mais do que ter à disposição, uma capacidade de computação disponível e flexível através de uma ligação à rede (Cooter, 2007). Todo o processamento, armazenamento, largura de banda[19], fiabilidade e segurança, são assegurados pela entidade prestadora dos serviços na Nuvem (*Cloud*), via internet. Em *datacenters* criados para o efeito. Permite serviços como: elasticidade, alocando e libertando recursos, conforme as necessidades. Por exemplo, uma loja web durante o ano utiliza apenas os recursos standards, mas ao prever que na época do Natal vai ter um pico de solicitações, nessa altura solicita um aumento dos recursos para o serviço, pagando apenas o valor correspondente por esse período de atividade (W. Van Den Bent & Van Der Steeg, 2012).

"Estamos vivenciando a transição do modelo atual de computação, com servidores isolados e aplicações monolíticas , para o modelo de computação em nuvem e softwares componentizados, oferecidos como serviços. É uma transformação que vai afetar o futuro da sociedade e como profissionais devemos estar conectados com o assunto (Turion, 2009).

[19] Medida da capacidade de transmissão de um determinado meio, conexão ou rede, determinando a velocidade a que os dados são transmitidos

Como qualquer outra tendência no mundo das TI,o *Cloud Computing* tem as suas vantagens e desvantagens nas mais diversas aplicações, quer no apoio ao negócio, pesquisa, desenvolvimento aplicacional ou para alguma finalidade específica.

Vantagens do Cloud Computing

O mais evidente benefício da utilização da computação em nuvem sistemas é o ROI [20]*(return of investment)*. devido à redução dos custos operacionais e de manutenção e operacionais relacionados com o, software e infraestrutura, que são pagos apenas enquanto são utilizados. Isto é bastante benéfico principalmente para empresas startup [21]que não tem que desviar capital para investir em equipamentos e serviços de administração. Para os utilizadores finais, a possibilidade de ter serviços e dados armazenados na nuvem, sempre disponíveis em qualquer lugar e momentos, via internet, através de vários dispositivos, como portáteis, smartphones, etc, é outra vantagem a ter em conta. A concentração de serviços e infraestruturas de TI em grandes *datacenters* também oferece uma considerável otimização em termos de afetação de recursos e eficiência energética, o que eventualmente pode levar a menos impacto sobre o meio ambiente (Buyya, Vecchiola, & Thamarai, 2013).

Desvantagens do Cloud Computing

Para Barry Sosinsky, apesar das vantagens que os serviços na nuvem apresentam é preciso ter em conta algumas desvantagens, como: Privacidade e Segurança - Apesar do grande esforço para garantir padrões fiáveis e reconhecidos internacionalmente, ainda não há muita clareza e consenso no que diz respeito a direitos e deveres entre as partes envolvidas num modelo de *Cloud Computing*. Ainda segundo Barry Sosinsky, outra desvantagem tem a ver com as possíveis necessidades de maior latência de rede como: transferência de dados elevada; aplicações que para a sua execução, necessitem de uma arquitetura e plataformas com hardware robusto (Sosinsky, 2011).

Outro fator a considerar é a localização. Cada país tem as suas próprias regulamentações (com impacto nos serviços globais de muitas empresas), o que obriga a actuar de acordo com a legislação vigente em cada região (Catteddu & Giles , 2009).

[20] Relação entre o dinheiro ganho ou perdido através de um investimento
[21] Empresa jovem e inovadora em qualquer área ou ramo de atividade, que procura desenvolver um modelo de negócio escalável e repetível

Biblioteca Digital

A aplicação Biblioteca Digital, foi desenvolvida com o intuito de disponibilizar a quem recorre a uma biblioteca tradicional, livros e toda a informação inerente em formato digital num ambiente virtualizado.

Cada cliente da Biblioteca tem credenciais próprias, nome de utilizador e palavra passe, para ter acesso remoto a uma máquina virtual única e exclusiva para cada utilizador. Após ter iniciado sessão remota na sua máquina a aplicação Biblioteca Digital inicia automaticamente apresentado uma janela onde o utilizador devera introduzir as suas credenciais de acesso à aplicação. De seguida é apresentada a janela onde poderá efetuar as suas consultas e transferência do livro em formato digital para a sua máquina virtual. A aplicação disponibiliza também todo o registo de pesquisas que efetuou bem como o registo de entradas na aplicação.

A aplicação foi desenvolvida utilizando o software de programação, *Microsoft Visual Studio 2013*, na linguagem *ADO.NET*. Para a implementação da aplicação vai ser necessário a criação de máquinas virtuais, servidores e clientes que irá ser descritos nos capítulos seguintes.

Este conjunto de maquinas virtuais é composto por:

- Um servidor como o nome DC, que irá ser o controlador de domínio e que disponibiliza também um servidor de DHCP.
- Um servidor com o nome BD1, que irá ser o servidor onde a base de dados da aplicação está alojada.
- Um servidor hipervisor com o nome HyperV1, onde irão ser criadas as maquinas virtuais clientes que irão executar a aplicação Biblioteca Digital.
- Uma máquina virtual com o nome DPTTI, onde o administrador da aplicação eo modo de acesso remoto faz a inserção, consulta, gestão de dados e utilizadores da aplicação.
- Uma máquina virtual com o nome Cliente01, onde o cliente faz a consulta e transferência dos seus livros.

Aplicação Biblioteca Digital

A aplicação é composta por 4 quadros

Figura 1 - Janela de autenticação

Botão Iniciar Sessão – Através dos dados introduzidos na caixa de texto do utilizador e password, faz uma consulta à base de dados verificando se o registo existe na mesma. O código poderá ser consultado no anexo login.xaml no bloco 35-88.

Botão Sair – Permite sair da aplicação. O código poderá ser consultado no anexo login.xaml no bloco 89-93.

Figura 2 - Janela Principal em modo de administrador - pesquisa e inserção de dados

Nesta janela quando o utilizador tem permissões de administração sera efetuada tanto a pesquisa como a inserção de dados e acesso ao quadro de gestão de utilizadores.

Podemos iniciar a interpretação de cada campo deste quadro principal da seguinte forma:

Caixa de texto Pesquisa – insere-se o texto que se pretende pesquisar

Botão procurar – Faz a procura do que se pretende pesquisar através da caixa de texto Pesquisa. O código poderá ser consultado no anexo *AdminView.xaml* no bloco 249-246.

Secção de resultados da pesquisa :

Caixa de combinação livros encontrados – Contém uma lista dos livros encontrados na pesquisa. Pode-se escolher um para visualizar as suas informações nos controlos em baixo.

Nº de registos – Mostra a quantidade de registos encontrados.

Botão de registos Anterior – Navegação para o registo anterior. O código poderá ser consultado no anexo *AdminView.xaml* no bloco 576-588.

Botão de registos Próximo – Navegação para o próximo registo. O código poderá ser consultado no anexo *AdminView.xaml* no bloco 589-600.

Caixa de texto Referencia – Mostra a referência do livro selecionado.

Caixa de texto Titulo – Mostra o título do livro selecionado.

Caixa de Texto Autor – Mostra o autor do livro selecionado.

Caixa de combinação Editora – Mostra a editora do livro selecionado.

Caixa de texto Resumo do Livro – Mostra o resumo do livro selecionado.

Imagem da capa do livro – Mostra a imagem da capa do livro selecionado.

Botão carregar imagem da capa do livro – Serve para carregar a imagem referente ao livro para a base de dados. O código poderá ser consultado no anexo *AdminView.xaml* no bloco 601-648.

Botão novo – Cria um novo registo de um novo livro. O código poderá ser consultado no anexo *AdminView.xaml* no bloco 347-358.

Botão Gravar – Grava um livro novo ou atualiza um livro existente. O código poderá ser consultado no anexo *AdminView.xaml* no bloco 371-566.

Botão Cancelar – Serve para cancelar o modo de edição do botão novo ou botão gravar. O código poderá ser consultado no anexo *AdminView.xaml* no bloco 359-370.

Botão Transferir Livro – Serve para descarregar uma cópia do ficheiro PDF para o disco local. O código poderá ser consultado no anexo *AdminView.xaml* no bloco 678-721.

Botão Carregar Livro – Serve para carregar uma nova cópia do ficheiro PDF para a base de dados. O código poderá ser consultado no anexo *AdminView.xaml* no bloco 722-763.

Na secção do perfil do utilizador a interpretação de cada campo é efetuada da seguinte forma:

Caixa de texto Nome – Nome do utilizador que iniciou sessão.

Caixa de texto Categoria – Categoria de utilizador. Administrador tem acesso a tudo e o utilizador só tem permissão de pesquisar e ver os livros e descarregar o PDF dos mesmos.

Caixa de texto Ultima sessão – Data-hora da última vez que iniciou sessão.

Imagem da foto do utilizador – Foto do utilizador.

Botão Gerir utilizadores – Serve para abrir uma janela de administração dos utilizadores desta aplicação. O código poderá ser consultado no anexo *AdminView.xaml* no bloco 764-773.

Botão Terminar sessão – Serve para terminar sessão para poder iniciar sessão com outro utilizador. O código poderá ser consultado no anexo *AdminView.xaml* no bloco 567-575.

Botão Limpar histórico – Serve para limpar o controlo onde são carregadas as pesquisas efetuadas pelo utilizador. O código poderá ser consultado no anexo *AdminView.xaml* no bloco 659-677.

Caixa de texto Historico de pesquisa – Lista de pesquisas efetuadas pelo utilizador.

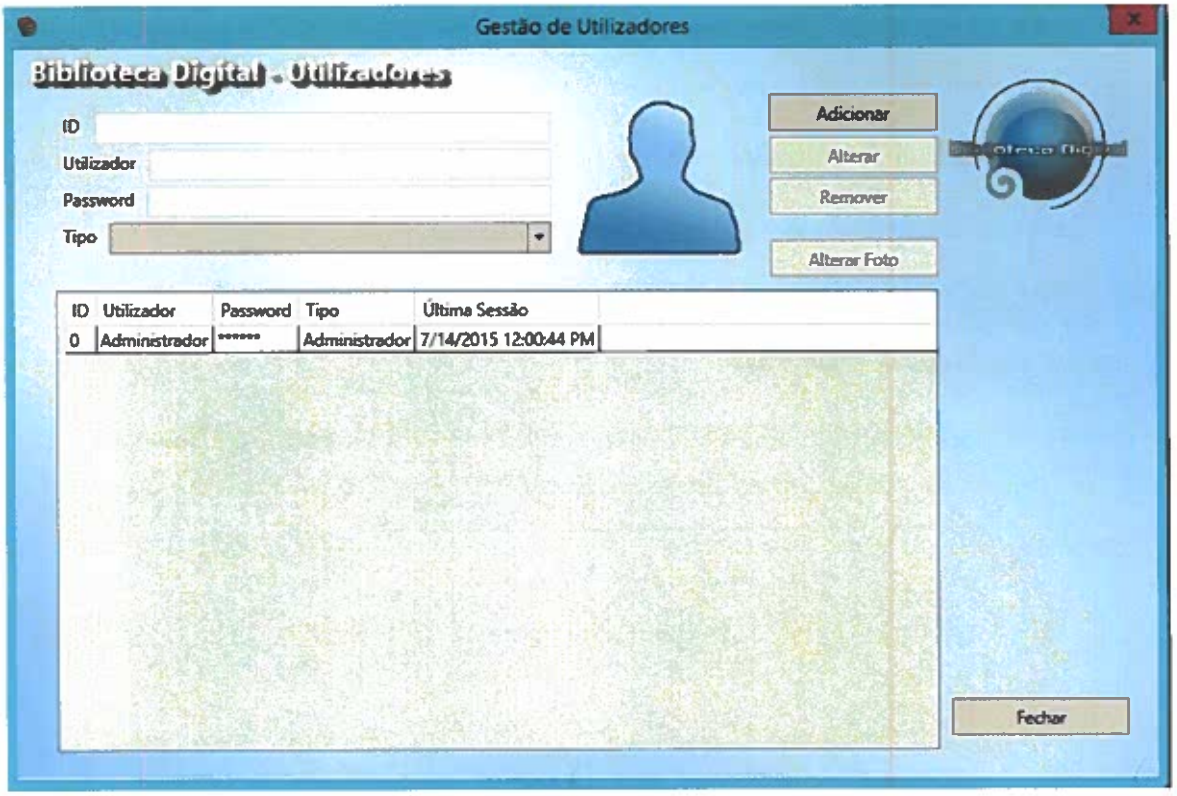

Figura 3 - Janela de Gestão de Utilizadores

Neste quadro da aplicação, é efetuada a gestão dos utilizadores. Podemos interpretar os campos da seguinte forma:

Caixa de texto ID – nº do utilizador

Caixa de texto utilizador – nome do utilizador

Caixa de texto password – palavra passe do utilizador

Caixa de combinação tipo – selecção do tipo administrador ou utilizador

Botão adicionar – Adiciona os dados de um novo utilizador através dos valores introduzidos nos controlos 'Utilizador', 'Password' e 'Tipo'. O código poderá ser consultado no anexo *ManageUsers.xaml* no bloco 189-235.

Botão alterar – Serve para atualizar os dados de um utilizador existente. O código poderá ser consultado no anexo *ManageUsers.xaml* no bloco 236-285.

Botão remover – Serve para eliminar um utilizador da base de dados. O código poderá ser consultado no anexo *ManageUsers.xaml* no bloco 286-329.

Botão alterar foto – Serve para alterar a foto que identifica o utilizador. O código poderá ser consultado no anexo *ManageUsers.xaml* no bloco 330-374.

Imagem da foto do utilizador – imagem do utilizador

Caixa da lista de utilizadores – mostra os utilizadores registados

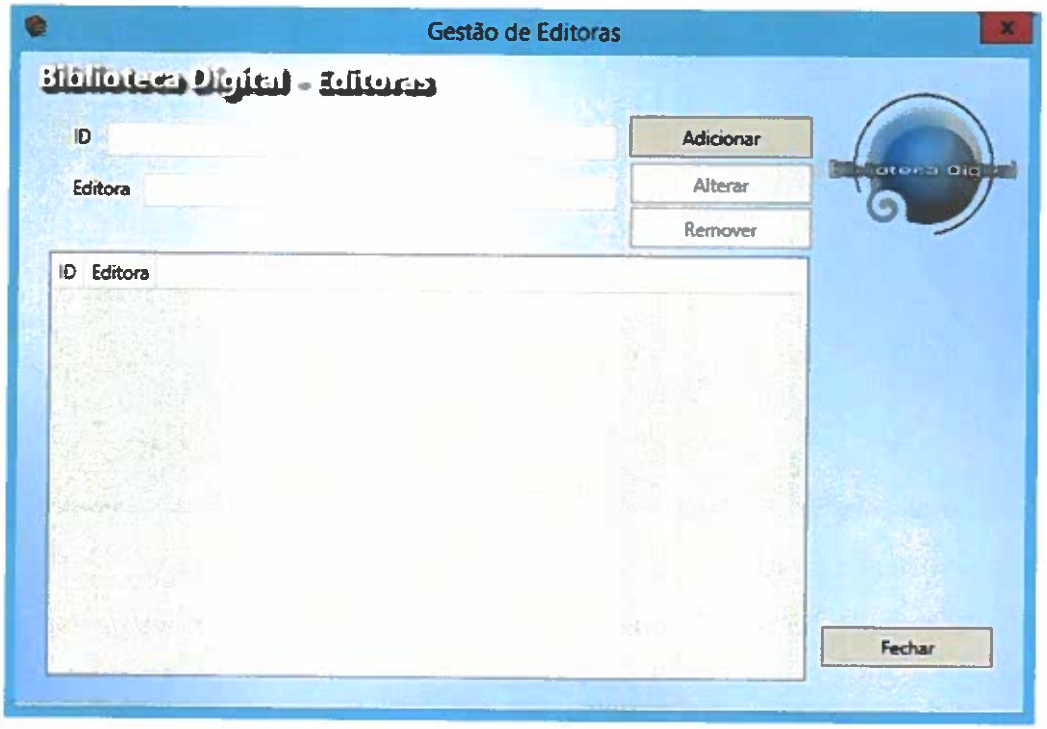

Figura 4 - Janela de gestão de editoras

Neste quadro da aplicação é onde é efetuada a gestão das editoras e podemos interpretar os campos de seguinte forma:

Caixa de texto ID – numero de identificação da editora

Caixa de texto editora – nome da editora

Botão adicionar – Adiciona os dados de uma nova editora através dos valores introduzidos no controlo 'Editora'. O código poderá ser consultado no anexo *ManagePublishers.xaml* no bloco 144-175.

Botão alterar – Serve para atualizar os dados de uma editora existente. O código poderá ser consultado no anexo *ManagePublishers.xaml* no bloco 176-220.

Botão remover – Serve para eliminar uma editora da base de dados. O código poderá ser consultado no anexo *ManagePublishers.xaml* no bloco 221-271.

Caixa da lista de editoras – apresenta todas as editoras registadas.

Figura 5 - Janela principal modo de utilizador

Todos os campos apresentados nesta janela são iguais à janela principal em modo de administrador em que foi mencionado anteriormente com a ressalva que o único campo onde o utilizador consegue editar é somente o da pesquisa, os restante estão somente em modo de leitura.

Criação de Máquinas Virtuais

Máquina virtual DC

Criação da máquina virtual no hipervisor *VMware Workstation* 11

Para a criação da máquina virtual que funcionará como controlador de domínio clica-se no ícone *"Create new virtual Machine"*.

Figura 6 - Consola VMWARE Workstation 11

Processo de criação da máquina virtual DC

Controlador de Dominio

Para a criação das máquinas virtuais no *vmware*, foram alterados alguns dos pontos padrão. Na janela indicada na figura 7, temos duas opções, *TYPICAL* (típica) e *CUSTOM* (personalizada), vamos escolher a segunda opção, pois teremos configurações bem definidas para nossa necessidade. De seguida após clicar em "*Next*", foi seleccionada a versão 11.0 do *VMWARE WORKSTATION* figura 8. Na figura 9, das opções possiveis,escolhemos: instalar a máquina virtual com uma imagem de disco *(iso)*[22].

[22] É uma *imagem* de CD, DVD ou BD de um sistema de ficheiros.

Figura 7 - Tipo de configuração

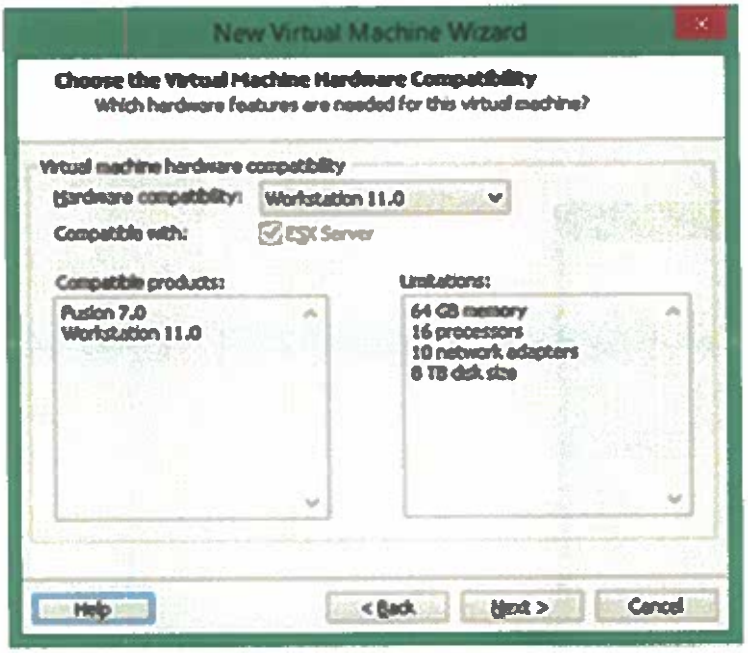

Figura 8 - Compatibilidade VMWARE

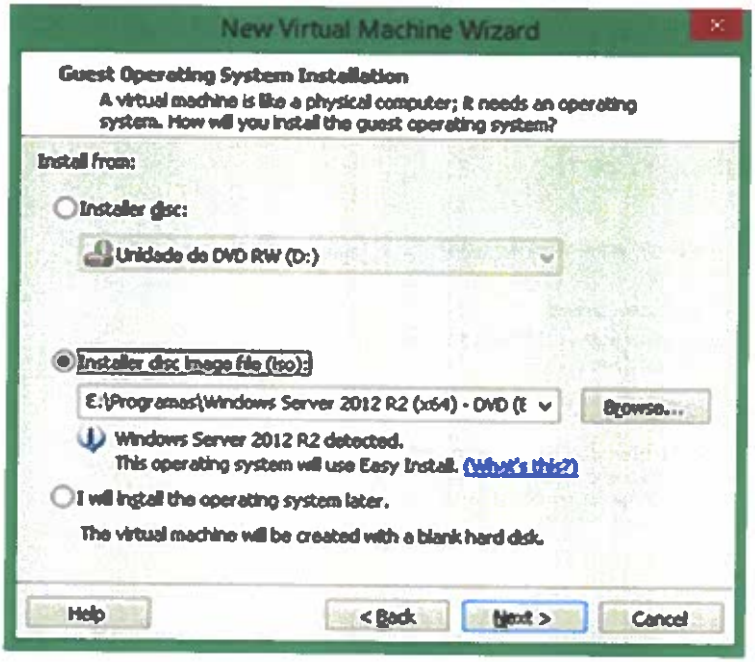

Figura 9 - Imagem para instalação

Foi atribuido o nome e localização para o *domain controller* (DC), figura 10. O tipo de firmware que permite escolher o dispositivo de arranque, neste caso *"BIOS"*, figura 11. Especificamos o numero de processadores e núcleos por processador para a máquina figura 12.

Figura 10 - Nome da máquina virtual

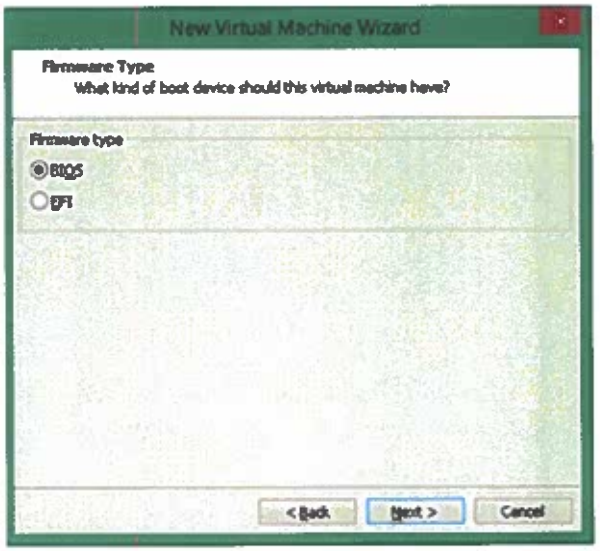

Figura 11 - Tipo de firmware

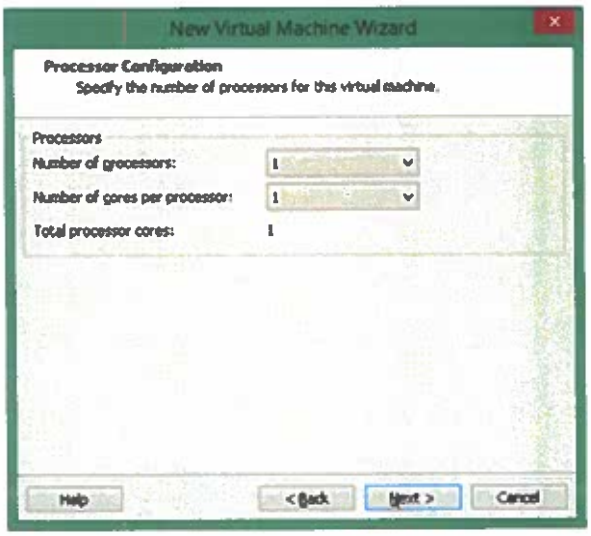

Figura 12 - Escolha de processadores

Foi introduzida a chave de produto para a versão do *windows server 2012 r2 datacentere password* de administrador, figura 13. Para as configurações de rede foi escolhido o *NAT*, por permitir acesso à rede externa, para actualizações, figura14. Seleccionamos 1024 MB de RAM para a memória da máquina, figura 15.

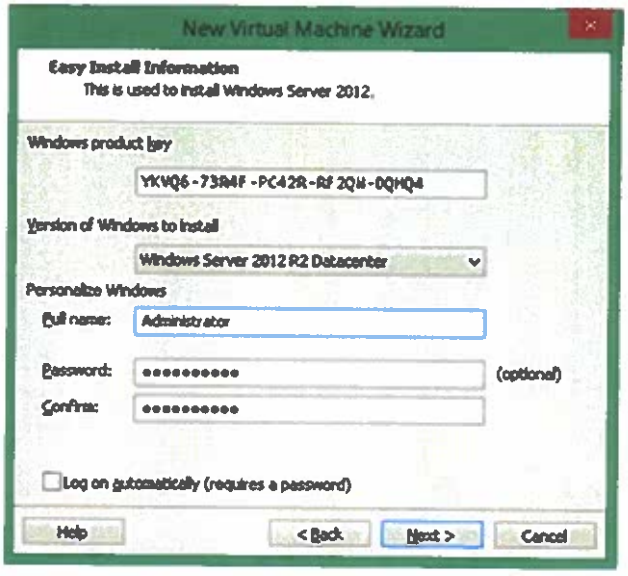

Figura 13 - Chave de produto

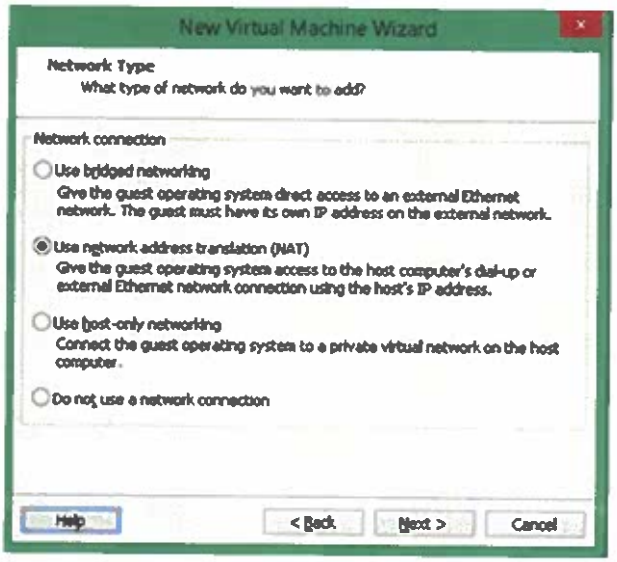

Figura 14 - Escolha de conectividade

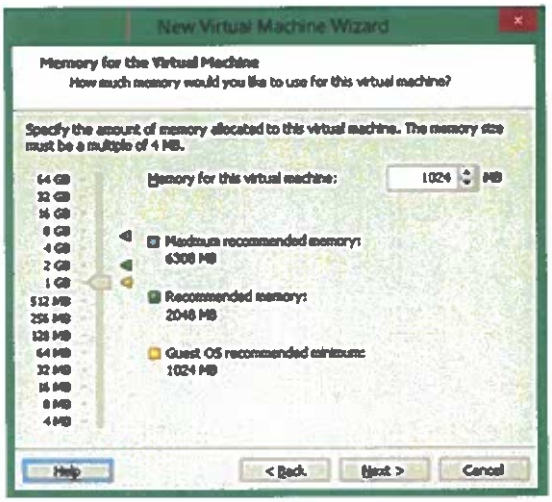

Figura 15 - Tamanho de memória

Configurámos o disco rígido virtual a ser usado, seleccionando a *opção "Create a new virtual disk"*, figura 16. De seguida deixamos a selecção por defeito para o controlador de disco, figura 17. O tipo de disco virtual, definimos como SCSI, figura 18.

Figura 16 - Disco virtual

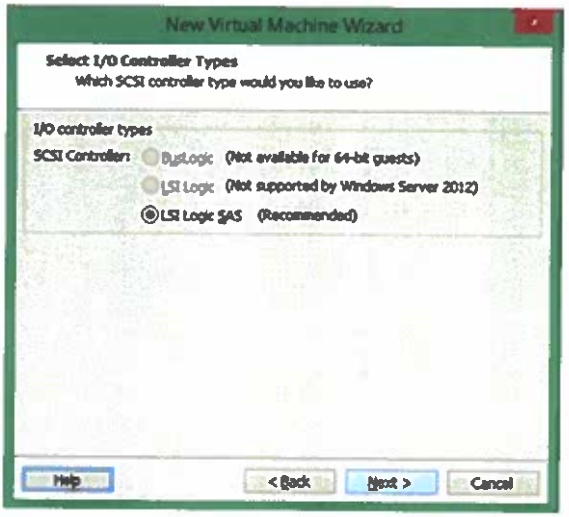

Figura 17 - Controladora SCSI

Figura 18 - Tipo de disco

Como tamanho de disco foram atribuídos 60 GB para a capacidade do disco rígido virtual. Foi seleccionada a opção *"Split virtual disk into multiple files"*, para o caso de ser necessário usar live migration, figura 19. Indicámos o caminho para a localização do disco, figura 20. Por fim finalisámos a creação da máquina virtual, figura 21.

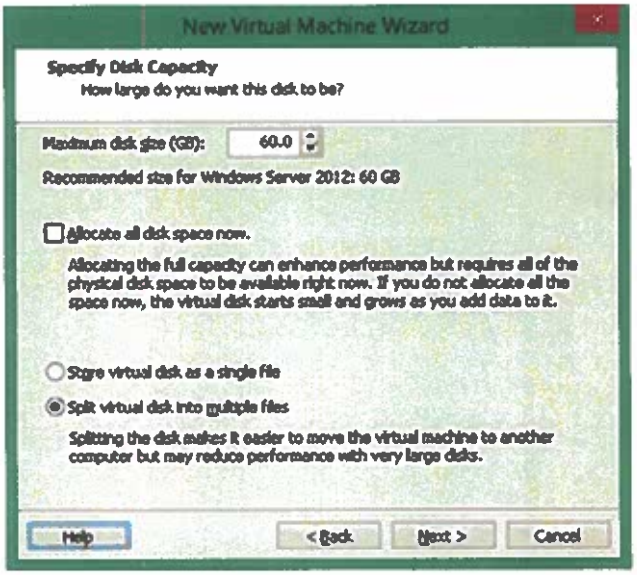

Figura 19 - Tamanho do disco virtual

Figura 20 - Localização do disco virtual

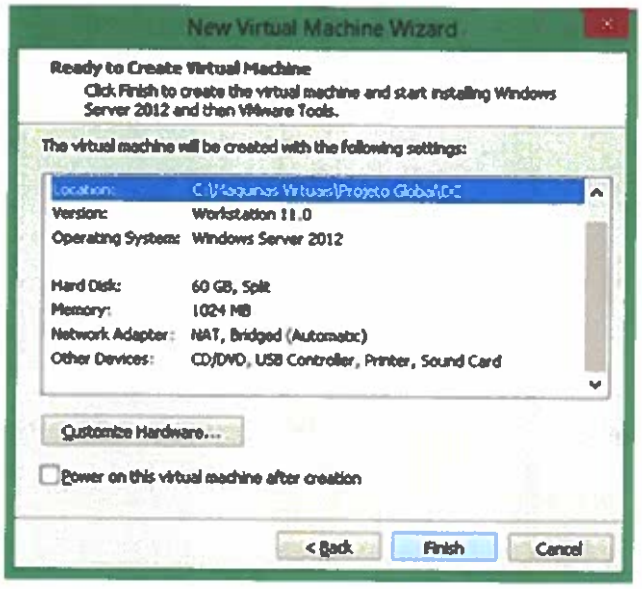

Figura 21 - Fim de configuração

Após a criação e instalação da máquina virtual e respectivas actualizações, procedemos à promoção a controlador de domínio, na consola do *Server Manager*, figura 22, clicando em "*Add roles features*".

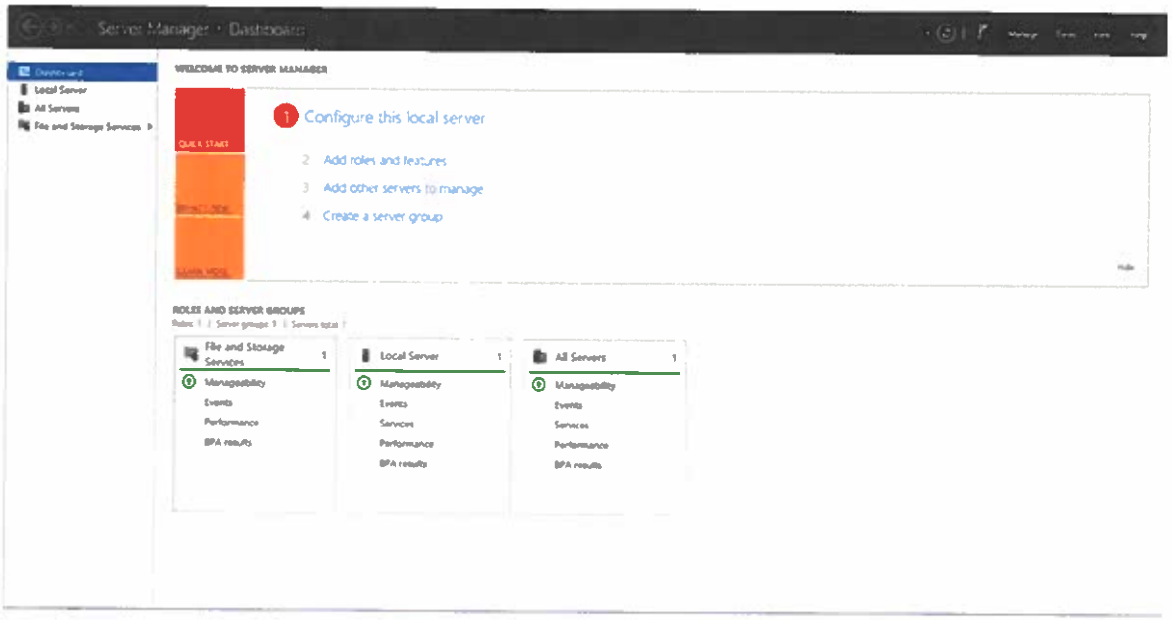

Figura 22 - Consola de gestão de servidor

Na figura 23, selecciona-se *Role-Based*.

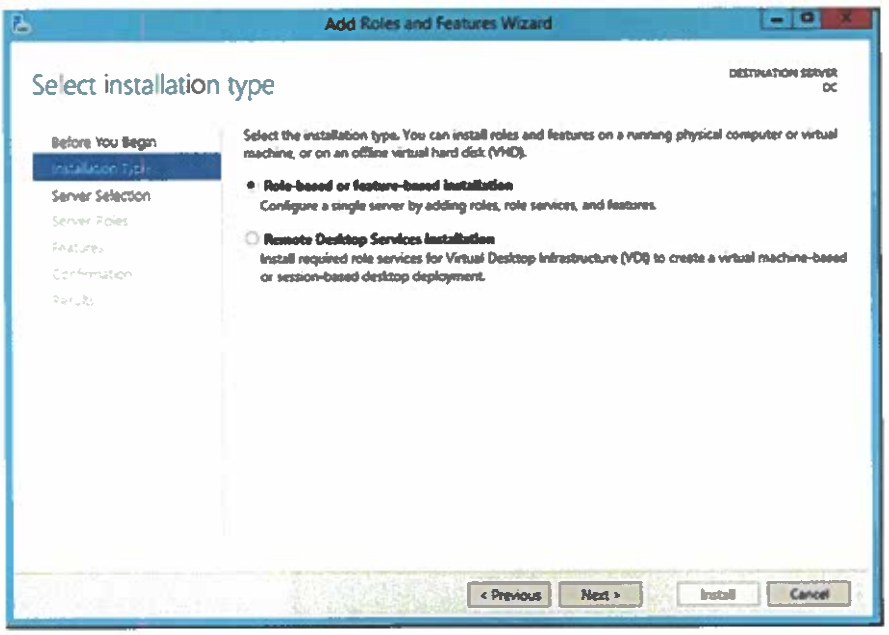

Figura 23 - Seleção de funcionalidade

Na selecção do servidor, escolhe-se a máquina que queremos para promover a controlador de domínio, figura 24.

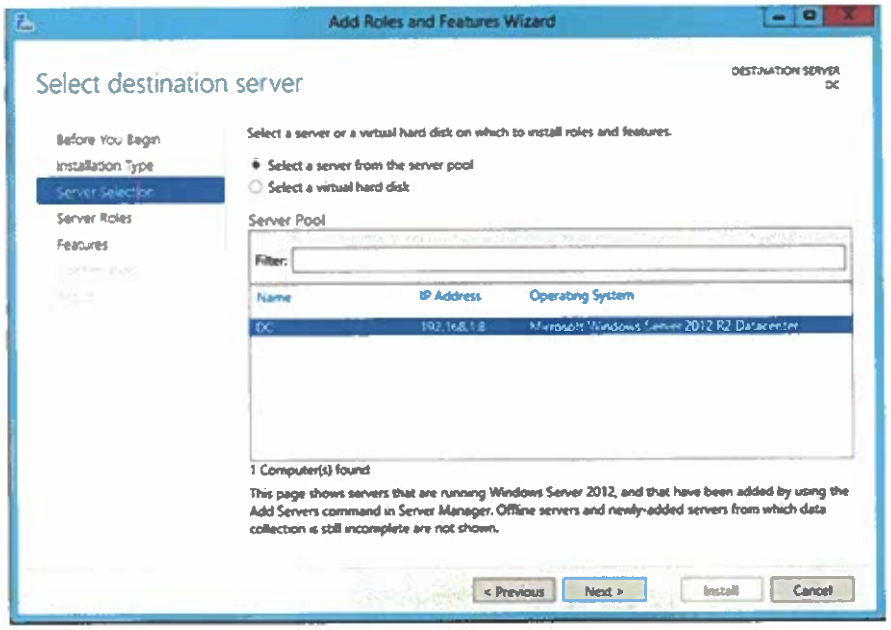

Figura 24 - Escolha de servidor

Activa-se a role *Active Directory Domain Services*, figura 25.

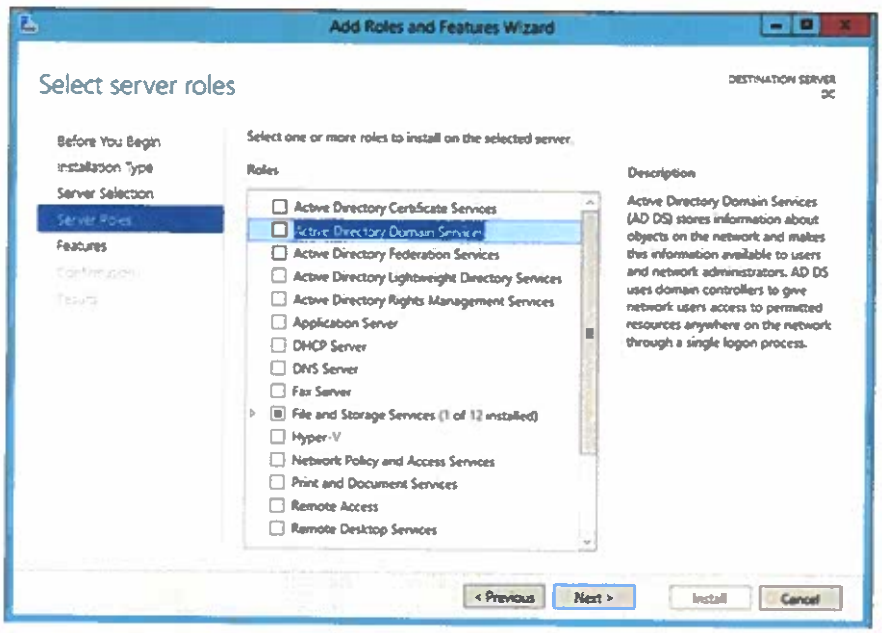

Figura 25 - Seleção de função

Depois do processo de instalação finalizar com sucesso, figura 26.

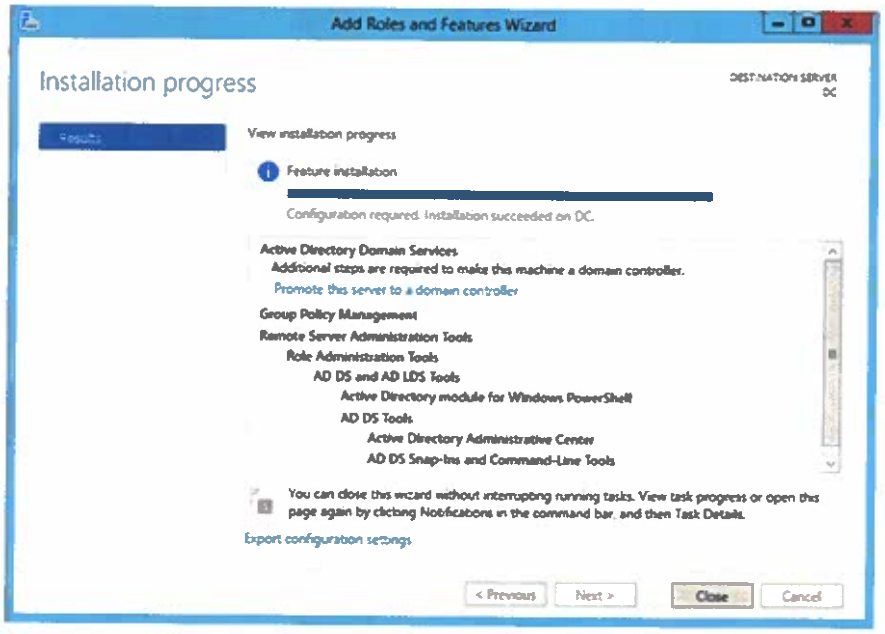

Figura 26 - Conclusão da instalação da função

Inicia-se a configuração do domínio, atribuindo-lhe um nome, figura 27.

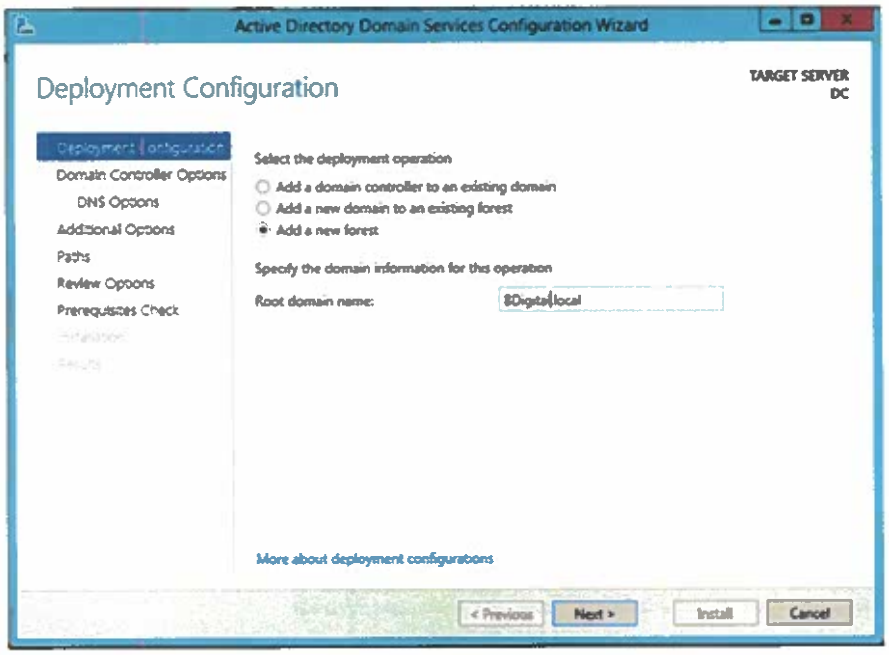

Figura 27 - Atribuição de nome ao domínio

Nas opções de configuração do controlador de domínio, atribui-se uma *password*, figura 28.

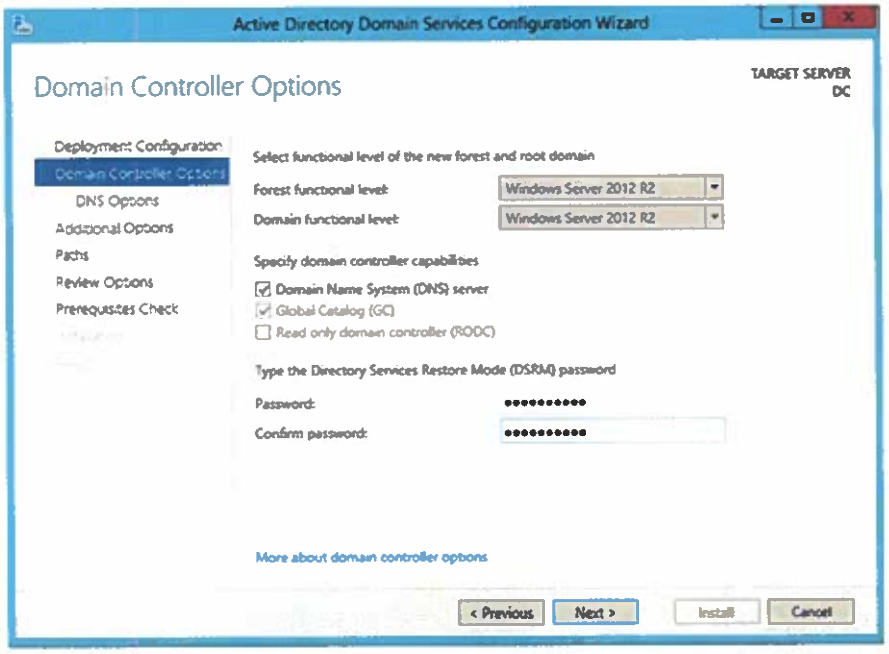

Figura 28 - Opções do controlador de domínio

Na figura 29, é verificado o *NetBIOS domain name*.

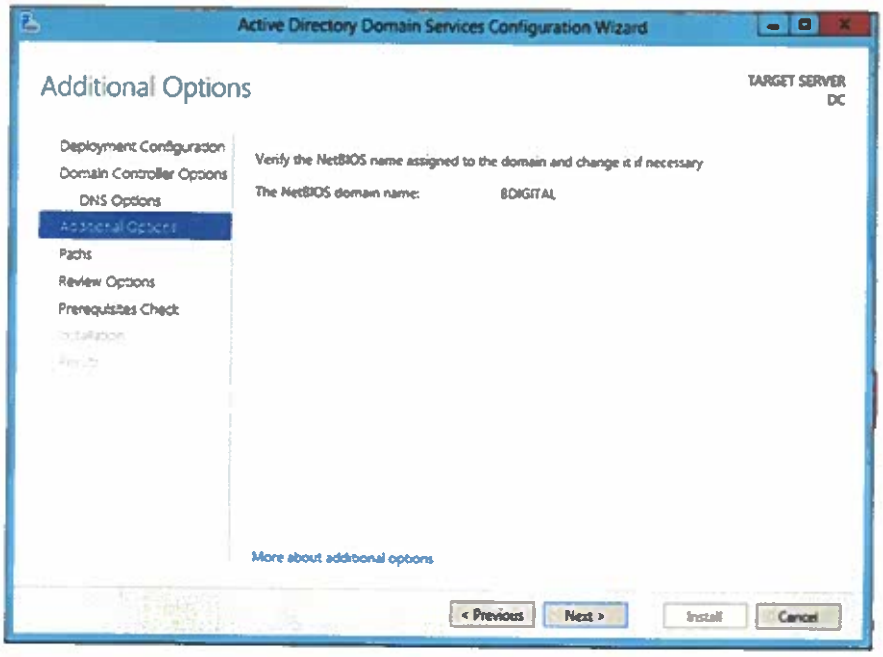

Figura 29 - Verificação de NetBIOS

Por fim verica-se a conclusão com sucesso dos pré-requisistos, figura 30.

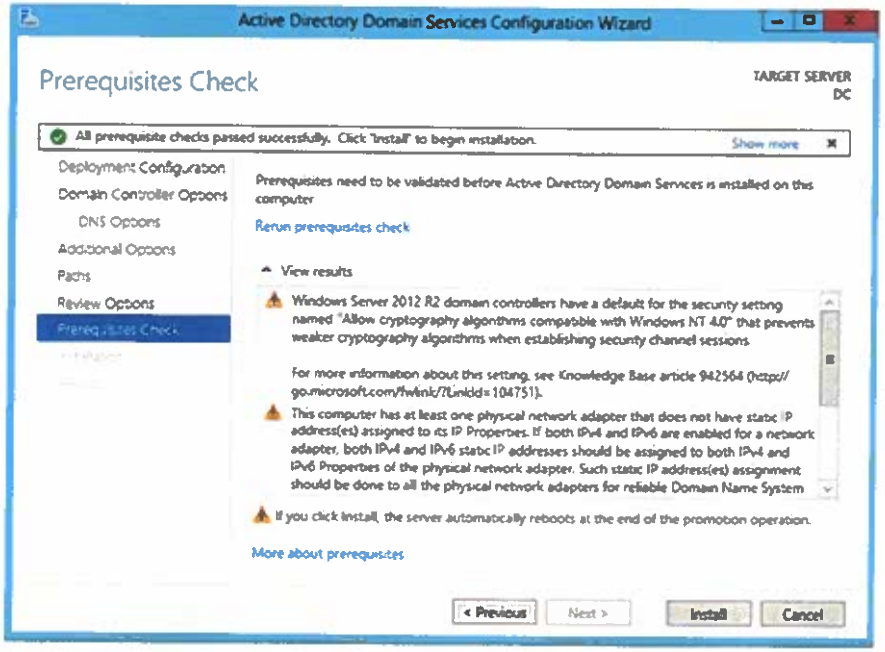

Figura 30 - Verificação de pré-requisitos

Criação de Unidades Organizacionais

Nas ferramentas do *Server Manager*, inicia-se o *Active Directory Users and Computers*, figura 31 e figura 32.

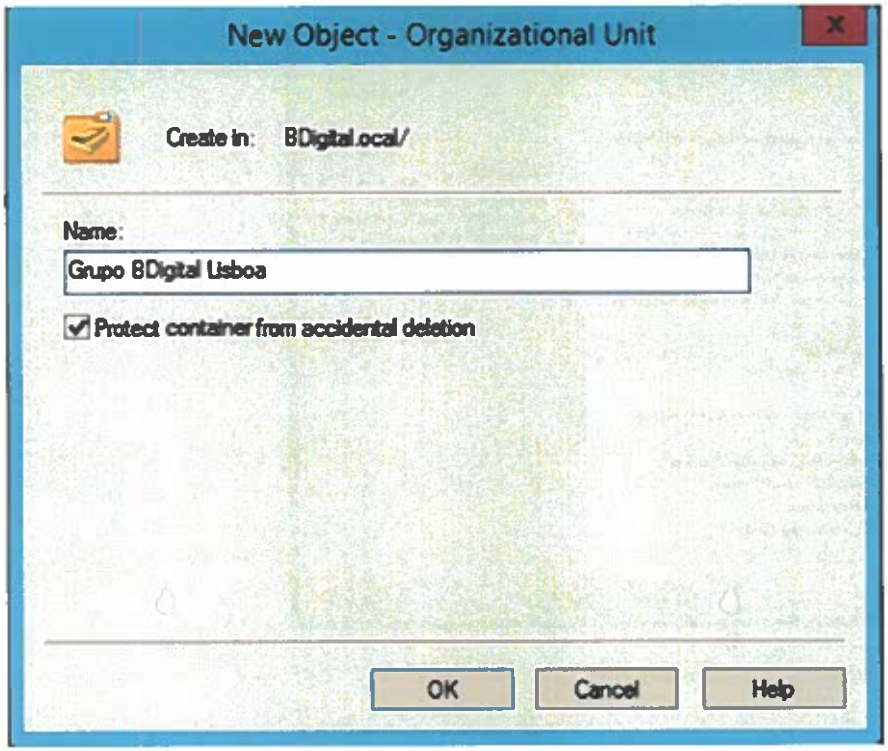

Figura 31 - Criação da unidade organizacional

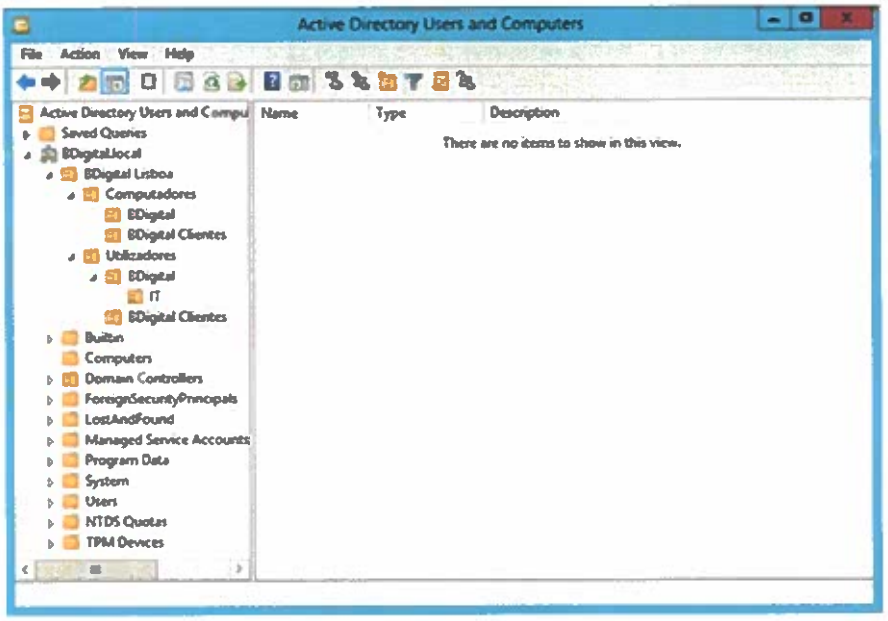

Figura 32 - Estrutura organizacional do Active Directory

Instalação do servidor de DHCP

No *Server Manager*, nas roles and features, marca-se a opção *DHCP Server*, para a instalação da funcionalidade, figura 33.

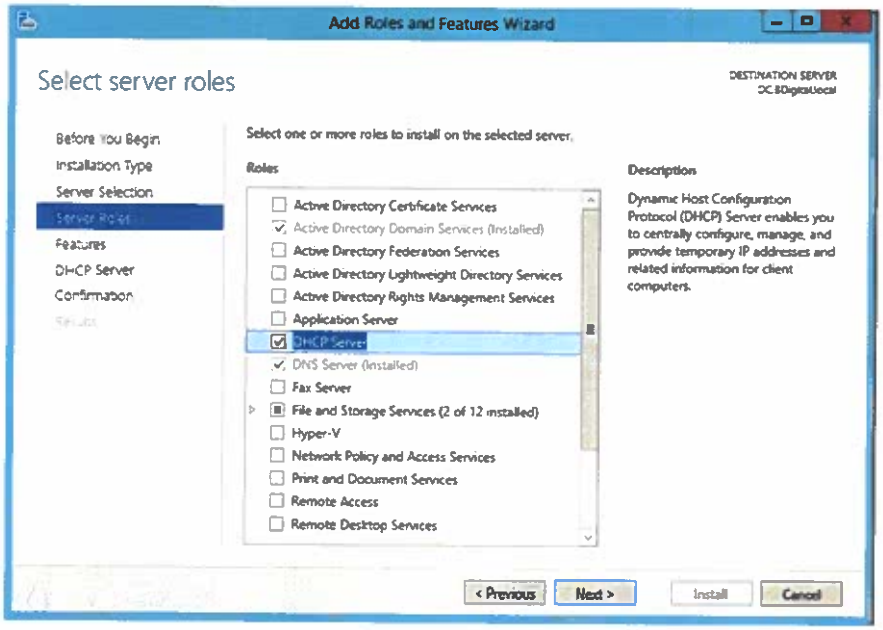

Figura 33 - Seleção da função de servidor de DHCP

Após a instalação concluída, na consola do *DHCP*, adiciona-se uma nova range de endereços *IPV4*, figura 34.

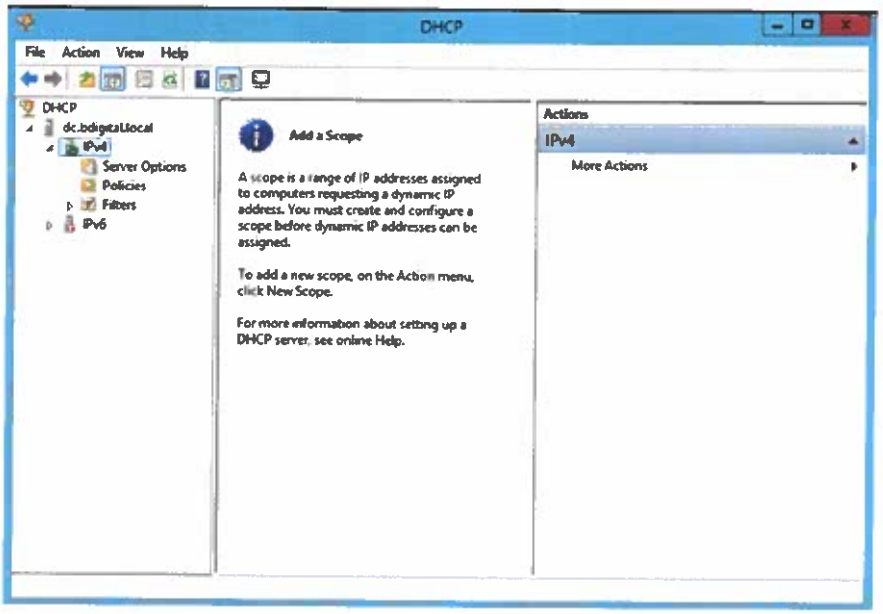

Figura 34 - Consola do servidor DHCP

Para criar a nova range de IP´s, cria-se um novo âmbito, figura 35.

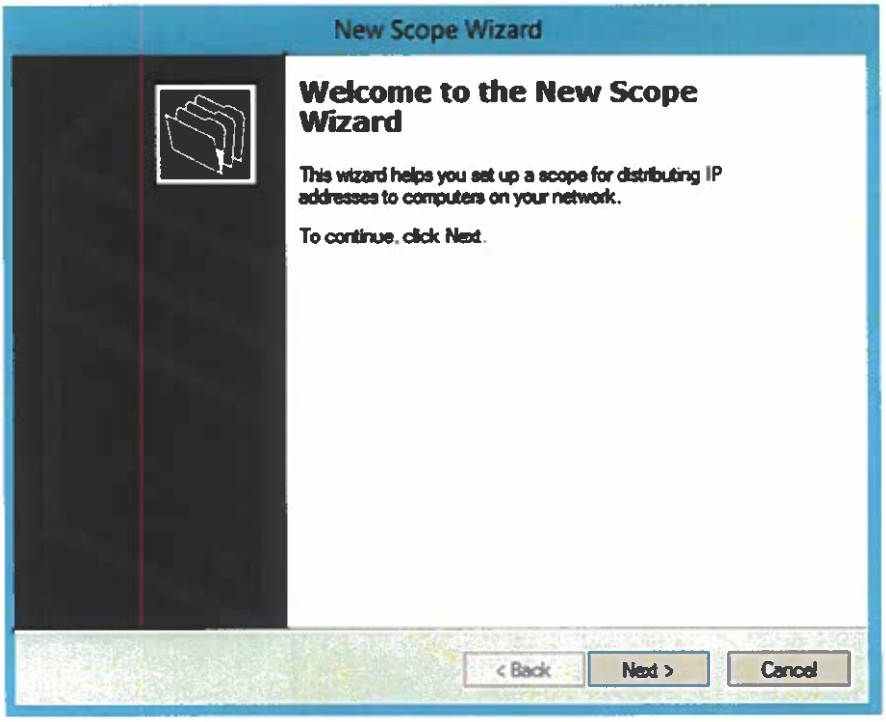

Figura 35 - Inicio de configuração para atribuição de IP´s

Atribui-se um nome, figura 36.

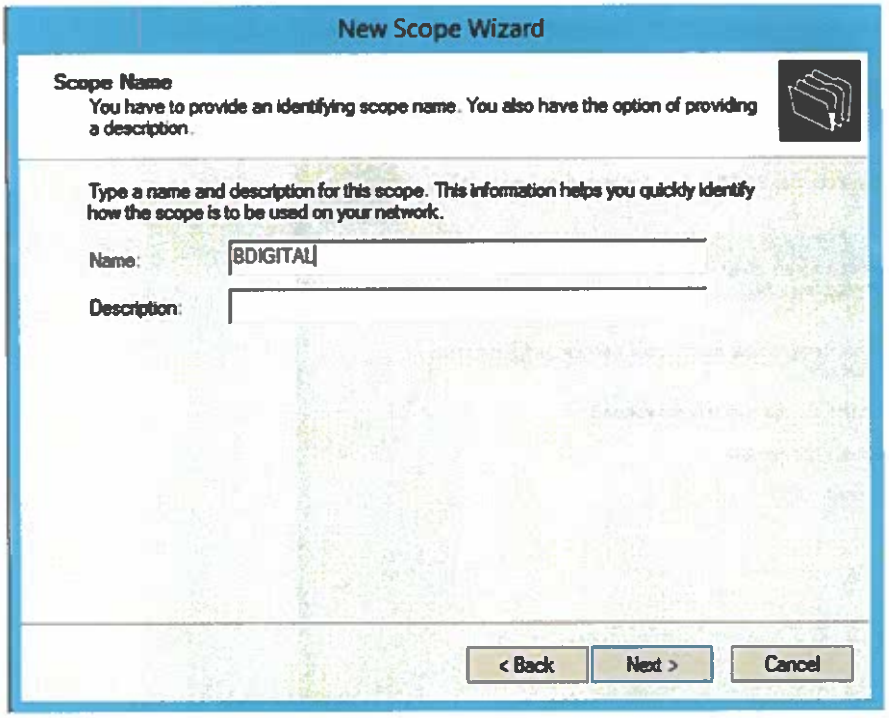

Figura 36 - Nome do Scope

Define-se a nova *range* de IP´s, figura 37.

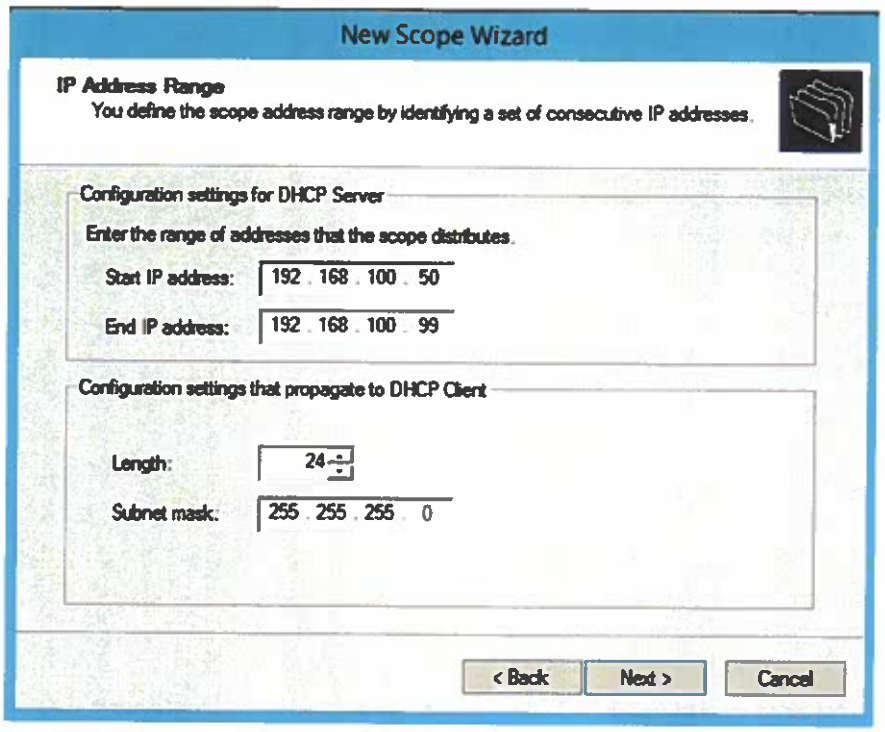

Figura 37 - Atribuição do intervalo de IP's

Configuram-se as opções do *DHCP*, figura 38.

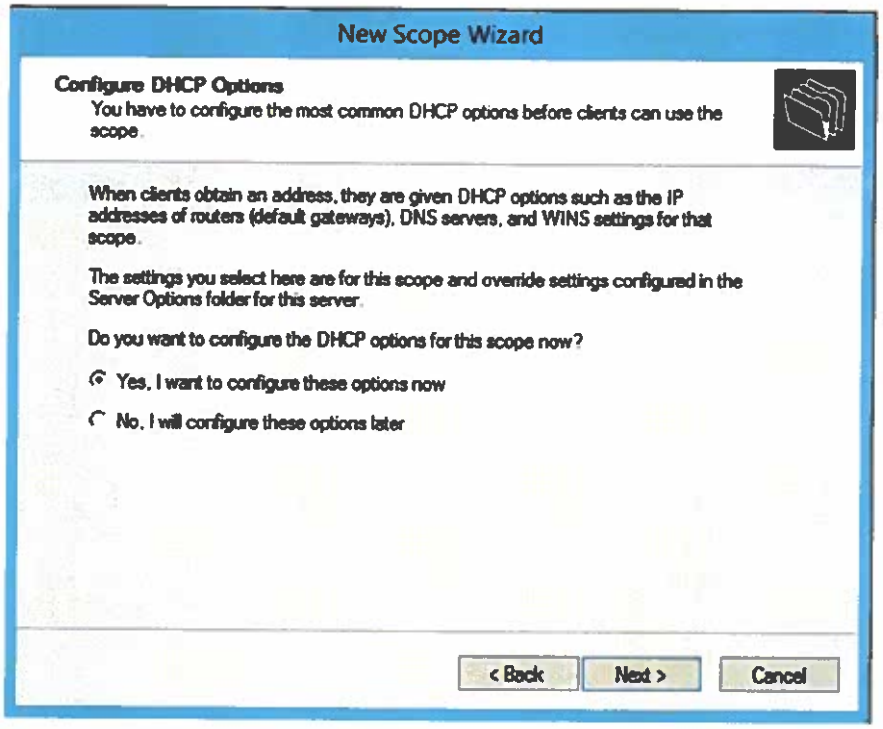

Figura 38 - Inicio de configuração de opções do scope

Nas opções do scope adiciona-se o IP do *router* para *default gateway*, figura 39.

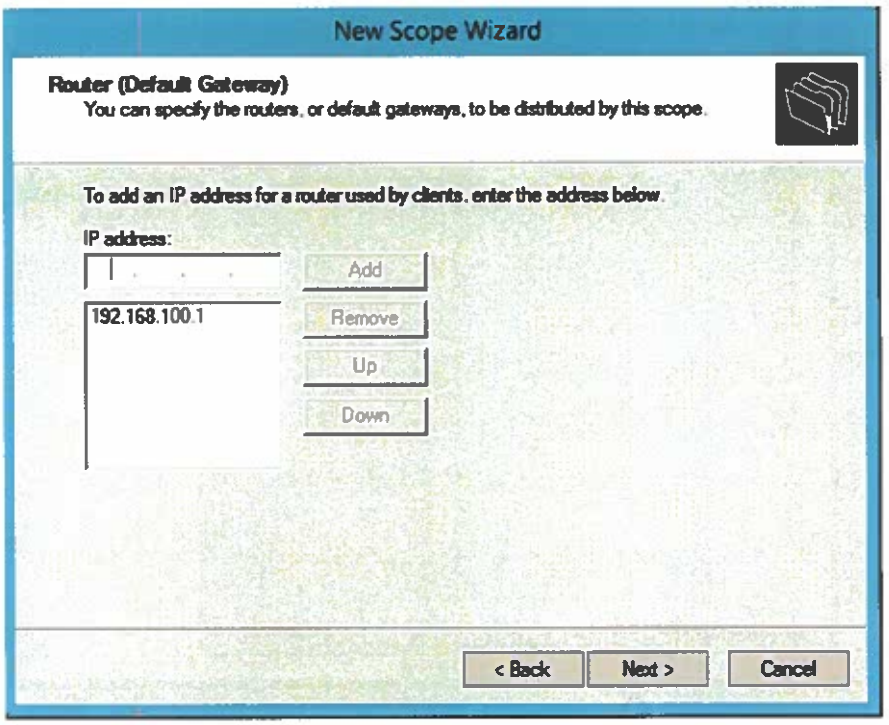

Figura 39 - Atribuição de IP para *gateway* padrão

Activa-se a *scope* do *DHCP*, figura 40.

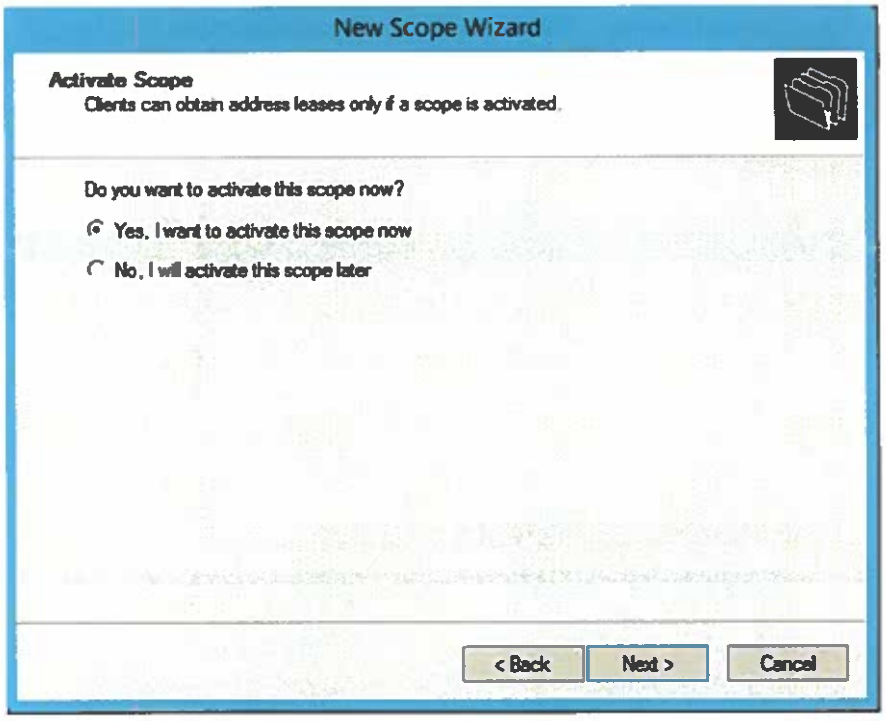

Figura 40 - Ativação do *scope*

Resultado da configuração bem sucedida, figura 41 para a *range* disponível de IP´s, e figura 42 para as opções do *scope*.

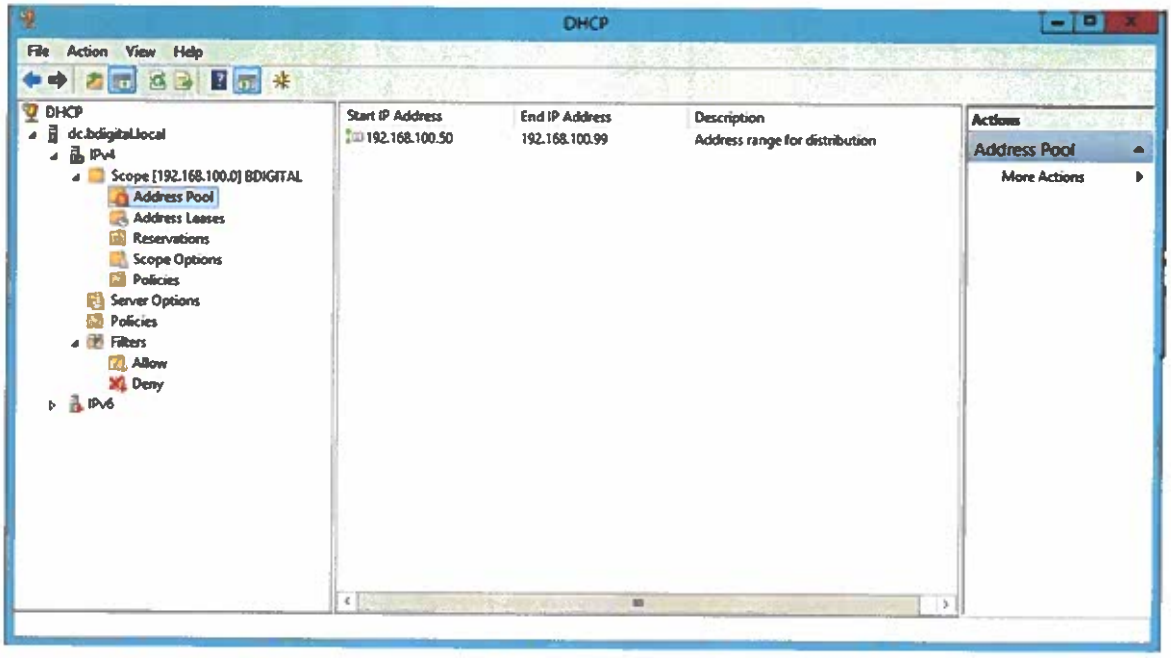

Figura 41 - Intervalo de IP´s

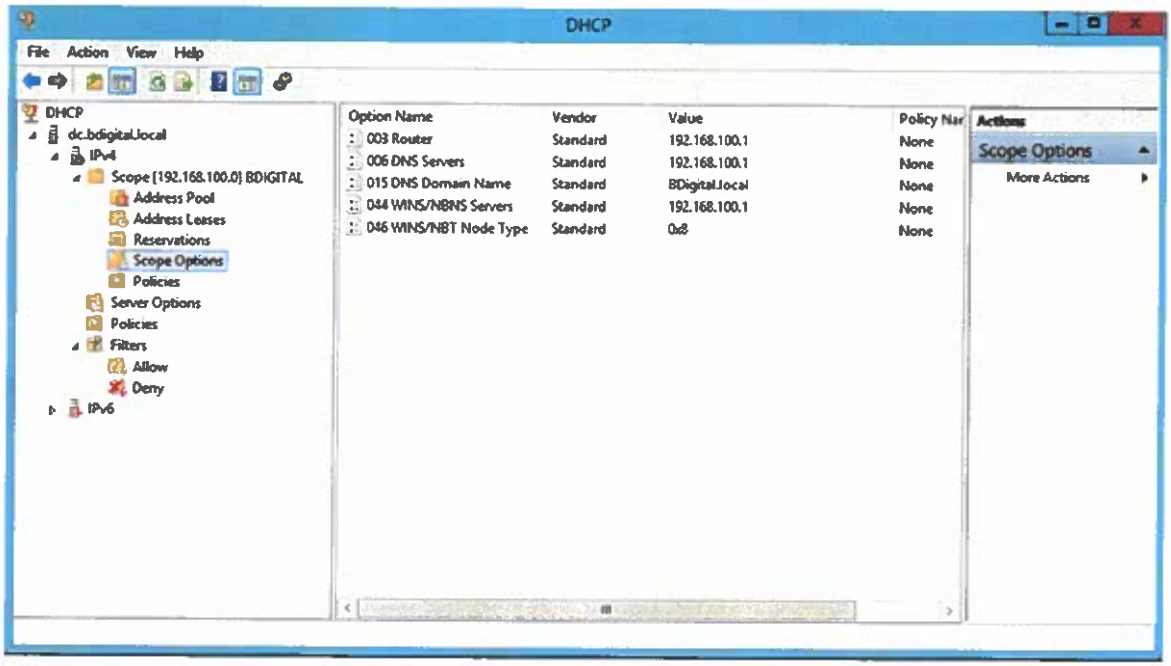

Figura 42 - Opções do *scope*

Máquina Virtual BD1

Processo de criação da máquina virtual BD1 – Servidor da Base de Dados

O processo de criação da máquina virtual BD1, é idêntico ao utilizado no capitulo anterior, com as seguintes diferenças:

- Memória RAM com 2048 MB
- Dois processadores com um núcleo

Após a instalação e configuração da máquina virtual BD1 e respectivas actualizações, procede-se à associação ao domínio, através da alteração das definições das propriedades do computador, como indicado na figura 43.

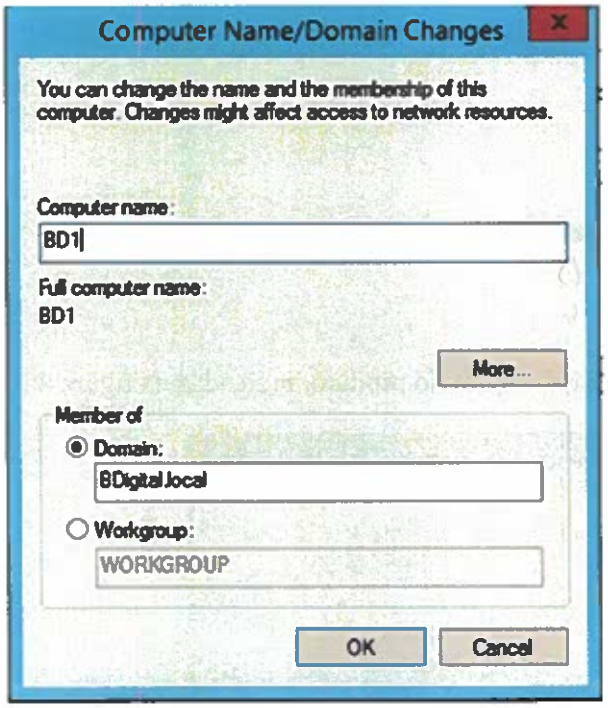

Figura 43 - Adicionar servidor ao domínio

Instalação e configuração do servidor de *SQL* na máquina virtual BD1

No quadro inicial do *SQL Server Installation Center*, referenciado na figura 44, inicia-se uma nova instalação isolada.

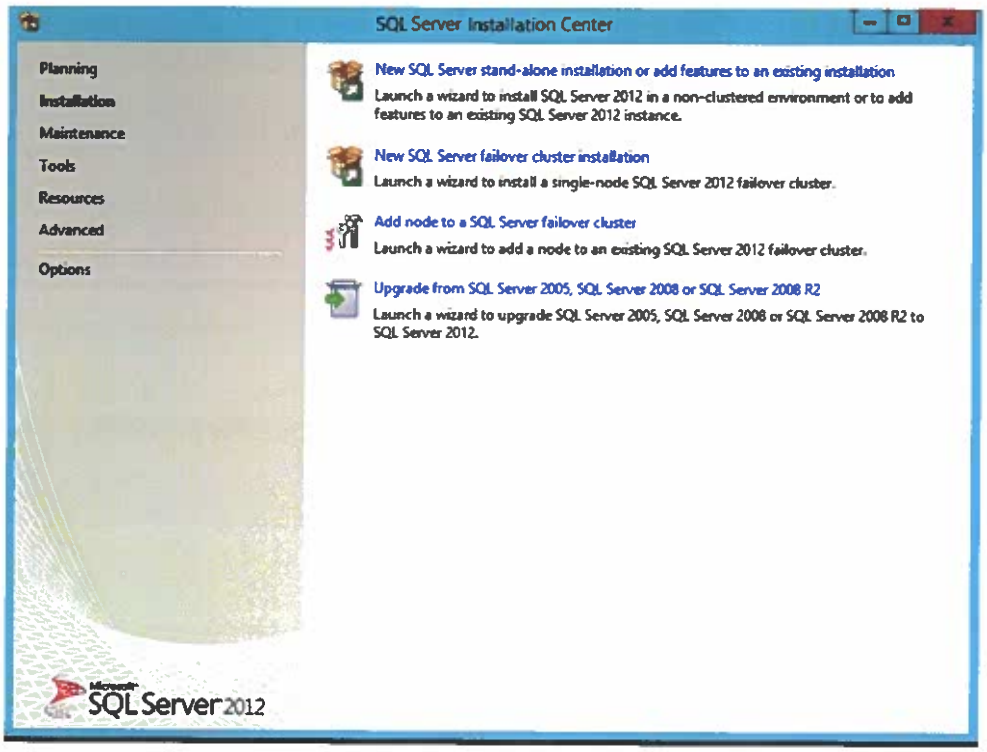

Figura 44 - Instalação de servidor de base de dados

Aconselha-se que sejam efectuadas as actualizações do produto, indicadas na figura 45.

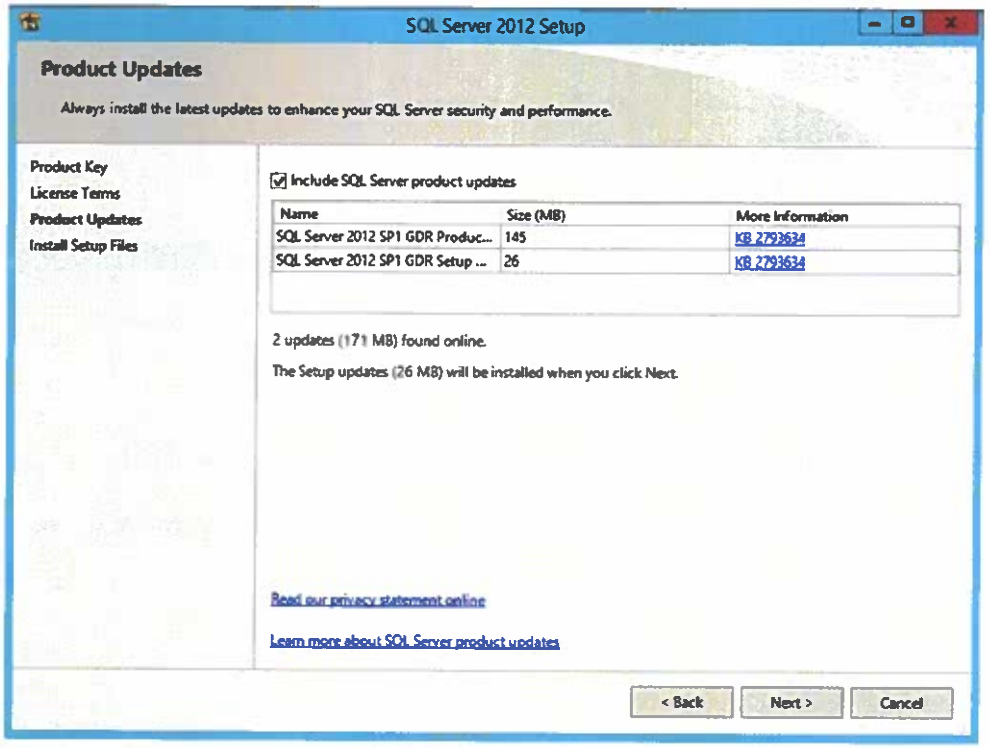

Figura 45 - Actualizações do *SQL*

Na janela do *Setup Role*, selecciona-se a opção *SQL Server Feature Installation*, figura 46.

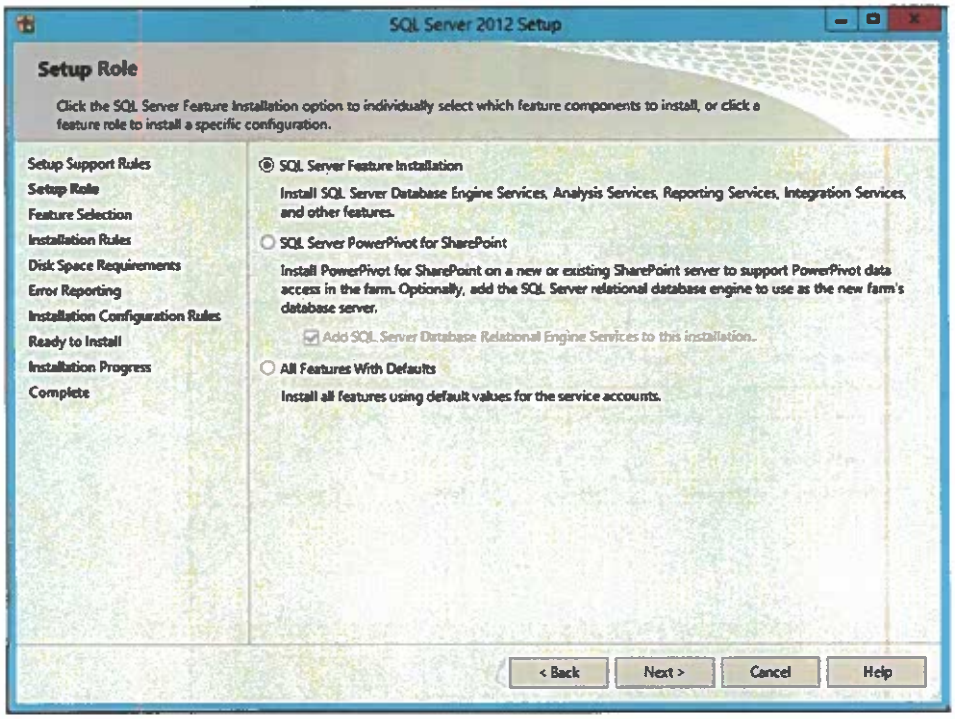

Figura 46 - Seleção da função de *SQL* a instalar

Na selecção activar todas as *features*, figura 47.

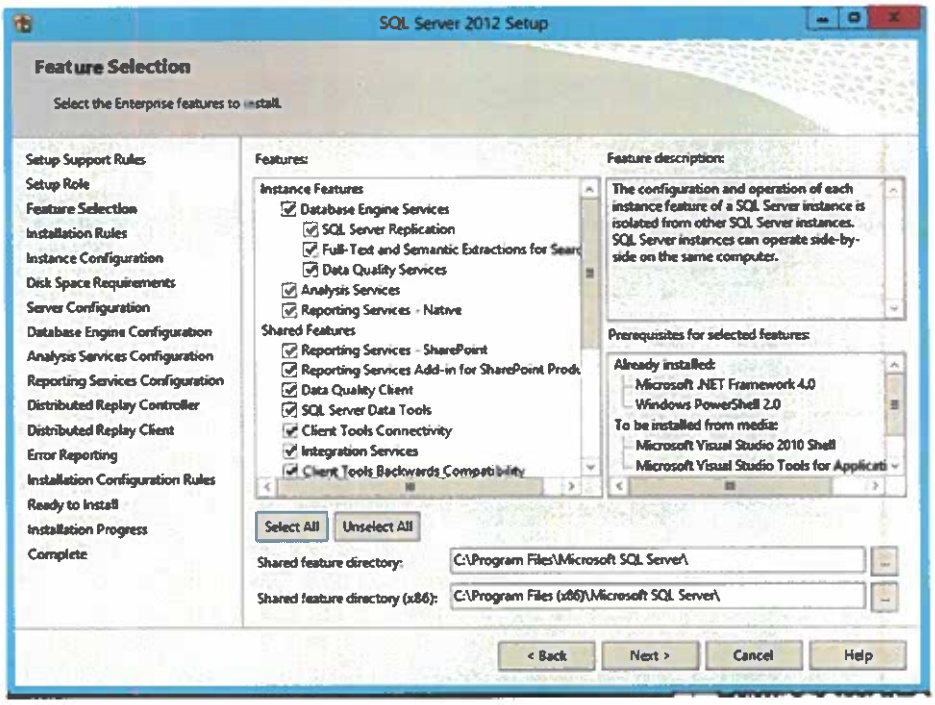

Figura 47 - Seleção de funcionalidades

Deixar como padrão o nome da instância, figura 48.

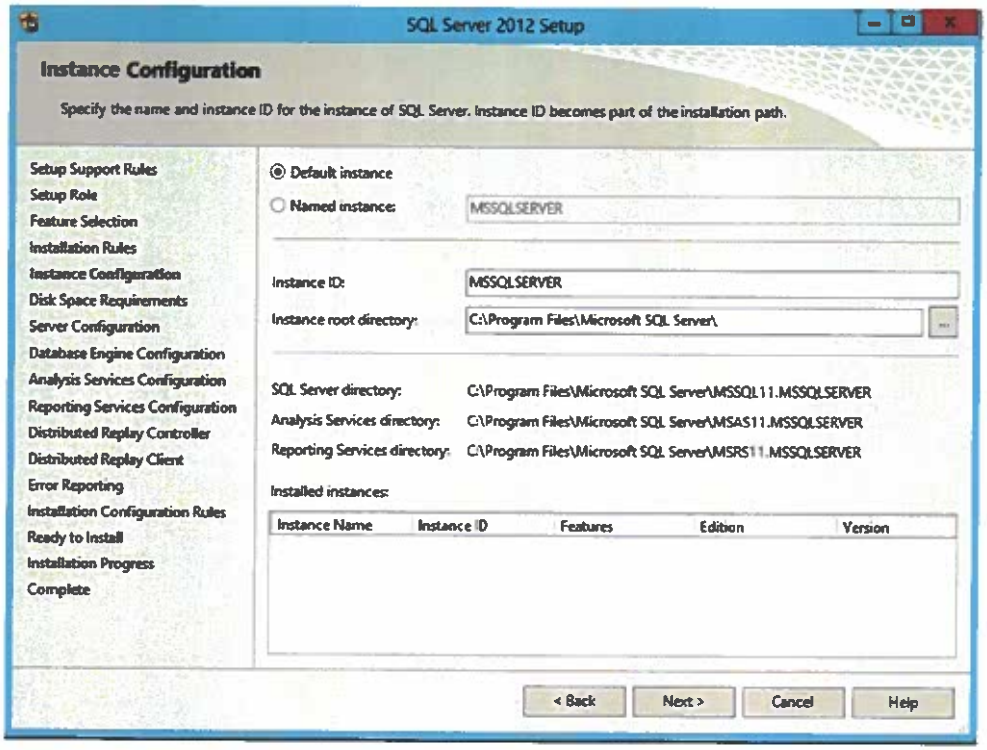

Figura 48 - Nome da instância

Especifica-se o utilizador com credenciais de administração para a base de dados, figura 49.

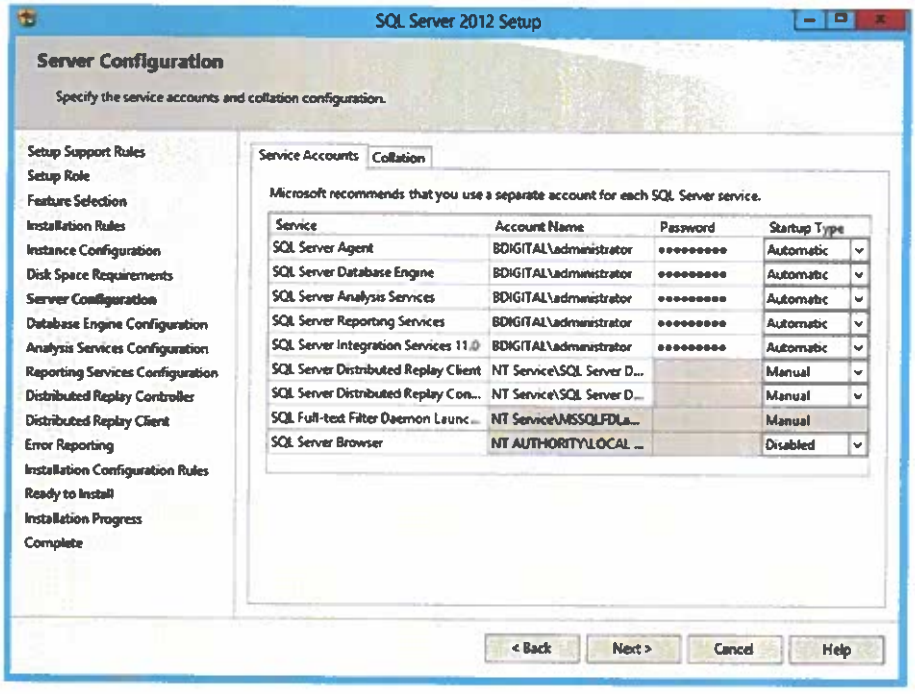

Figura 49 - Configuração de segurança

No modo de autenticação, selecciona-se o modo Windows, figura 50.

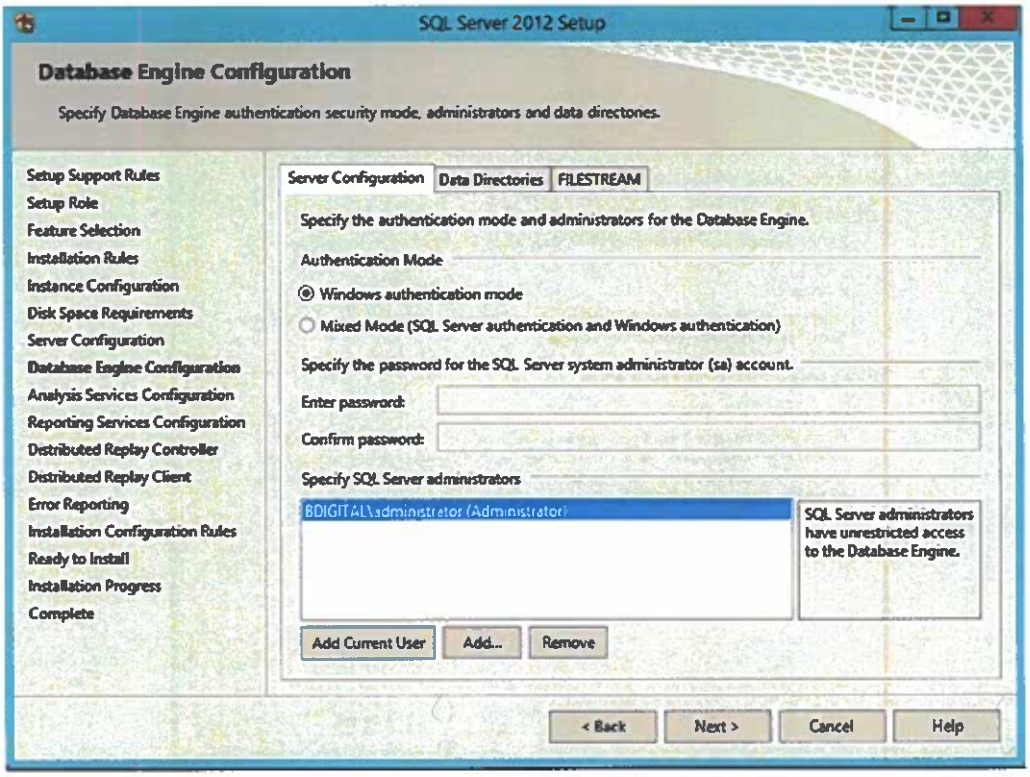

Figura 50 - Motor de base de dados

Figura 51, selecciona-se o modo multidimensional.

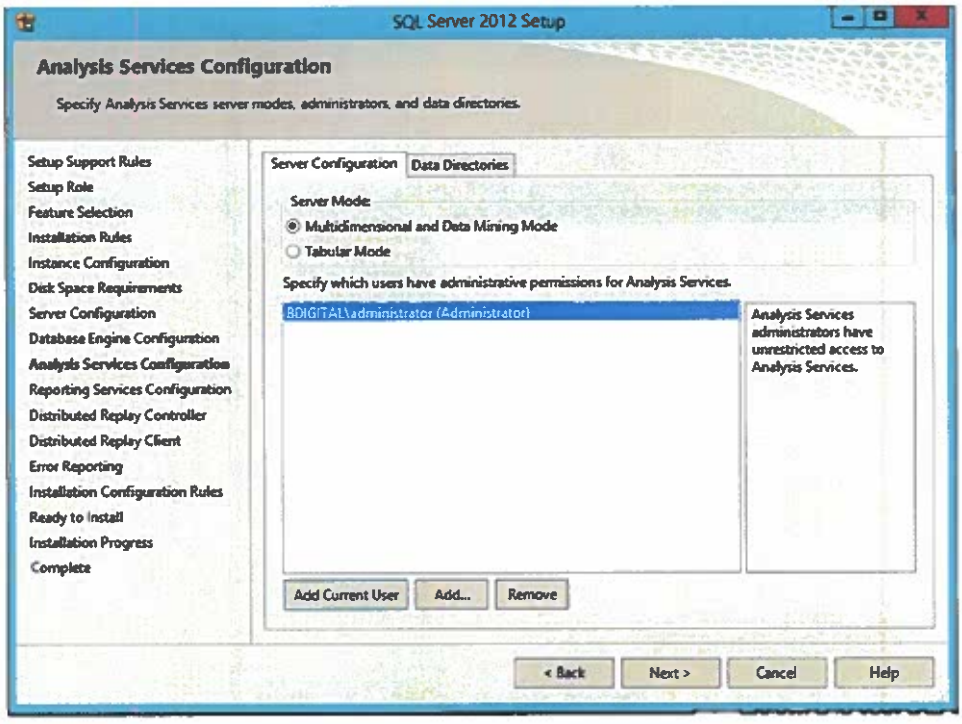

Figura 51 - Analisador de serviços

Mantém-se por padrão o tipo de serviços de configuração de reportação, figura 52.

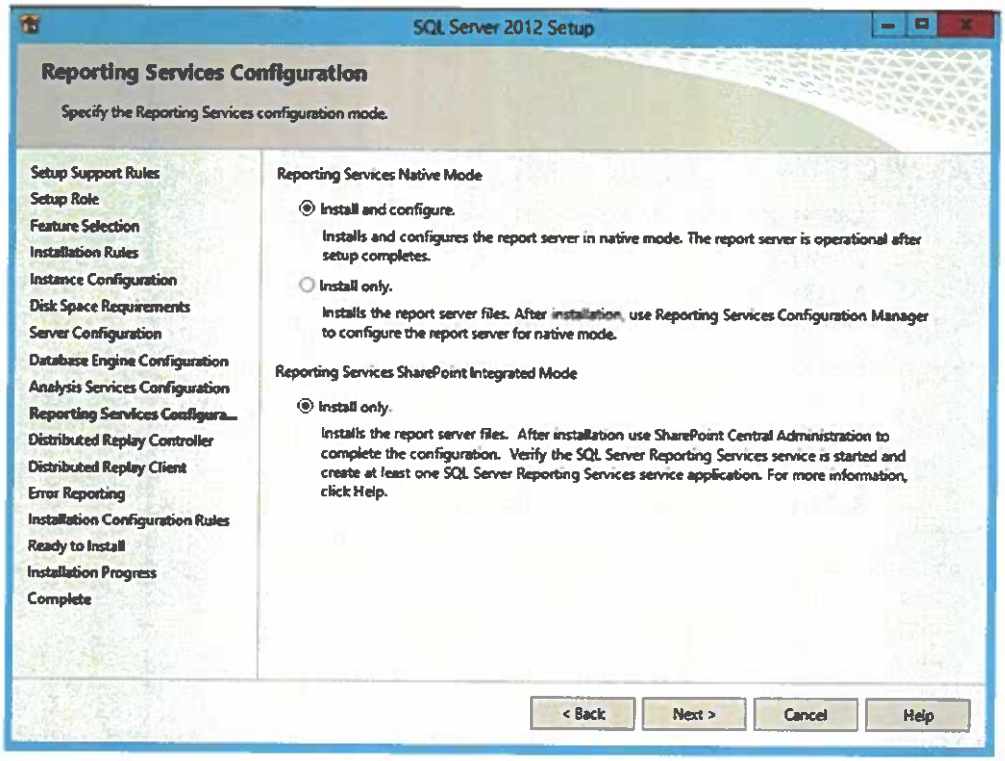

Figura 52 - Reportação de serviços

Adiciona-se o utilizador com permissões para o controlador de distribuição, figura 53.

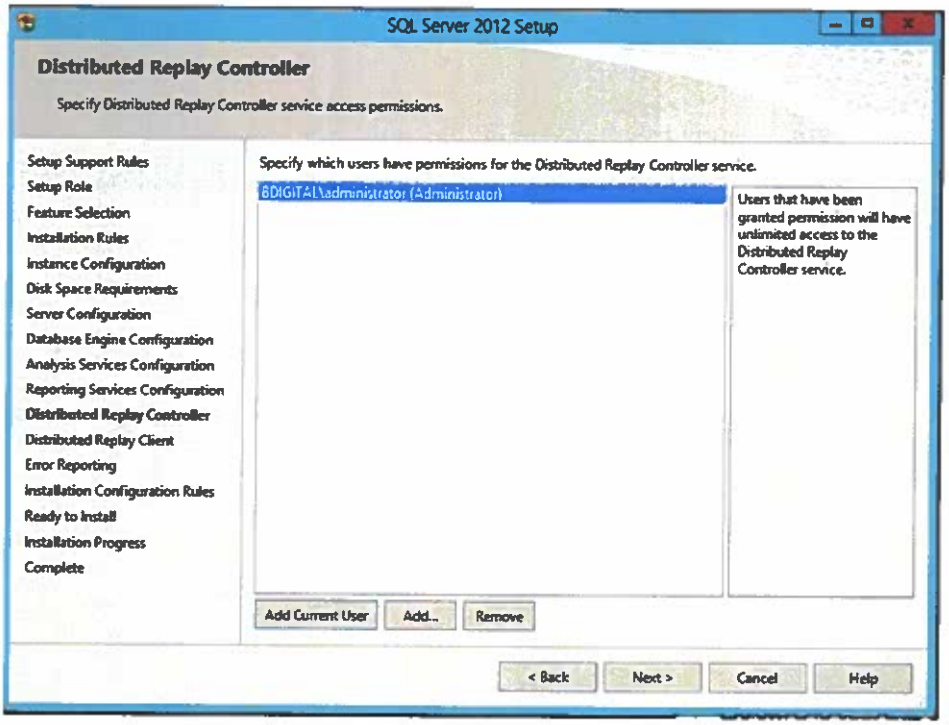

Figura 53 - Controlador de distribuição

Por fim conclui-se a instalação, figura 54.

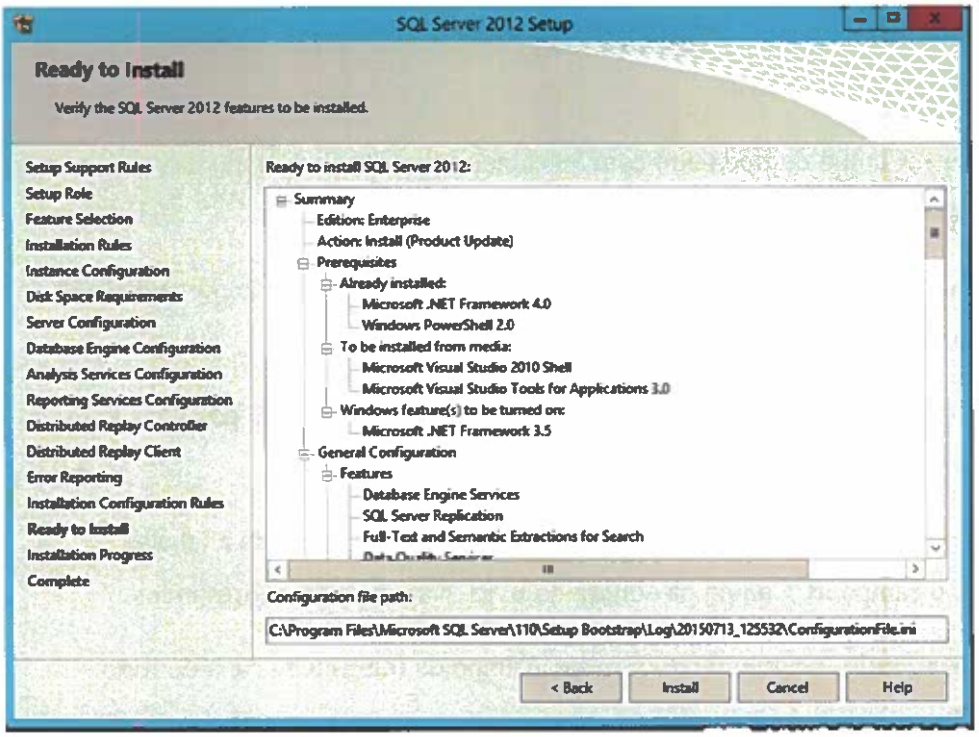

Figura 54 - Finalização da configuração

Diagrama da base de dados

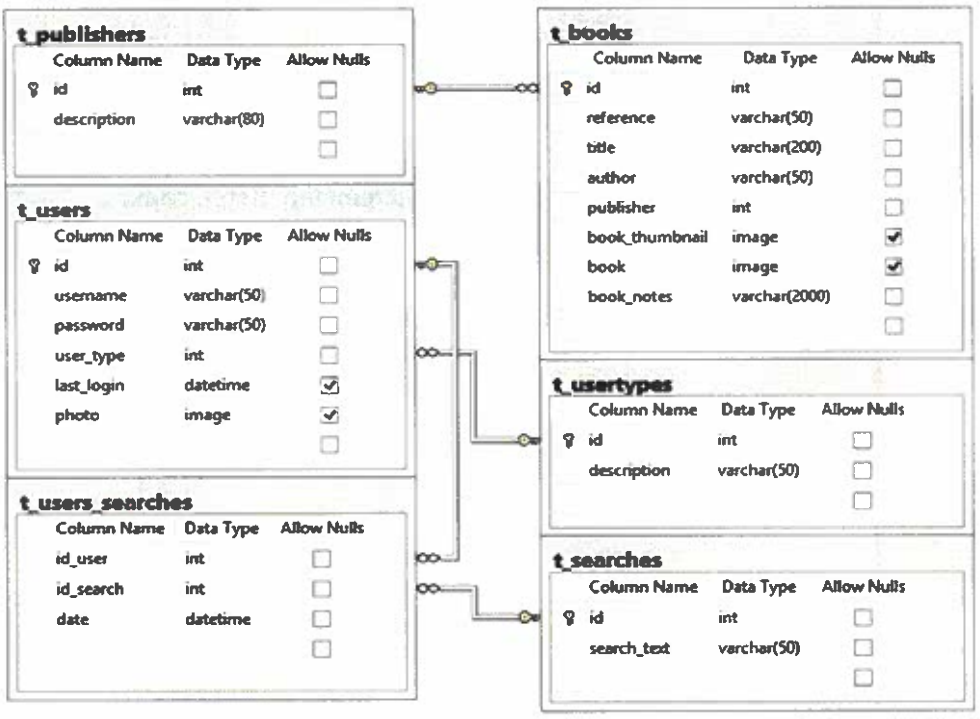

Figura 55 - Diagrama da base de dados da aplicação

Tabela t_books

Campo id – Campo chave do tipo número inteiro com auto incremento. Serve como indentificador inequívoco do registo.

Campo *reference* – Campo do tipo texto com 50 caracteres máximo. Campo da referência do livro. Não suporta valores nulos.

Campo title – Campo do tipo texto com 200 caracteres máximo. Campo do nome do livro. Não suporta valores nulos.

Campo *author* – Campo do tipo texto com 50 caracteres máximo. Campo do autor do livro. Não suporta valores nulos.

Campo *publisher* – Campo do tipo número inteiro. Chave estrangeira com a tabela t_publishers com o campo id. Campo da editora do livro. Não suporta valores nulos.

Campo *book_thumbnail* – Campo do tipo image. Campo da imagem da capa do livro. Suporta valores nulos.

Campo *book* – Campo do tipo image. Campo do ficheiro PDF do livro. Suporta valores nulos.

Campo *book_notes* – Campo do tipo texto com 2000 caracteres máximo. Campo do resumo do livro. Não suporta valores nulos.

Tabela t_publishers

Campo *id* – Campo chave do tipo número inteiro com auto incremento. Serve como indentificador inequívoco do registo.

Campo *description* – Campo do tipo texto com 80 caracteres máximo. Campo do nome da editora. Não suporta valores nulos.

Tabela t_users

Campo *id* – Campo chave do tipo número inteiro com auto incremento. Serve como indentificador inequívoco do registo.

Campo *username* – Campo do tipo texto com 50 caracteres máximo. Campo do nome de utilizador da aplicação. Não suporta valores nulos.

Campo *password* – Campo do tipo texto com 50 caracteres máximo. Campo da palavra-passe do utilizador da aplicação. Não suporta valores nulos.

Campo *user_type* – Campo do tipo número inteiro. Chave estrangeira com a tabela t_usertypes com o campo id. Campo do tipo de utilizador da aplicação. Não suporta valores nulos.

Campo *last_login* – Campo da data da última sessão iniciada pelo utilizador.

Campo *photo* – Campo da fotografia associada ao utilizador.

Tabela t_usertypes

Campo *id* – Campo chave do tipo número inteiro com auto incremento. Serve como indentificador inequívoco do registo.

Campo *description* – Campo do tipo texto com 50 caracteres máximo. Campo do tipo de utilizador. Não suporta valores nulos.

Tabela t_searches

Campo *id* – Campo chave do tipo número inteiro com auto incremento. Serve como indentificador inequívoco do registo.

Campo *search_text* – Campo do tipo texto com 50 caracteres máximo. Campo do texto pesquisado. Não suporta valores nulos.

Tabela t_users_searches

Campo *id_user* – Campo do tipo inteiro. Chave estrangeira com a tabela t_users com o campo id. Serve como identificador único juntamente com o campo id_search.

Campo *id_search* – Campo do tipo inteiro. Chave estrangeira com a tabela t_searches com o campo *id*. Serve como identificador único juntamente com o campo id_user.

Campo *date* – Campo da data da pesquisa efetuada pelo utilizador.

Máquina virtual *Hyper-V*

Processo de criação da máquina virtual *Hyper*-V1 – Servidor Hipervisor

O processo de criação da máquina virtual *Hyper*-V1, é idêntico ao utilizado no capitulo do controlador de dominio, com as seguintes diferenças:

- Memória *RAM* com 3072 MB
- Dois processadores com um núcleo

Instalação da role *Hyper-V*, figura 56.

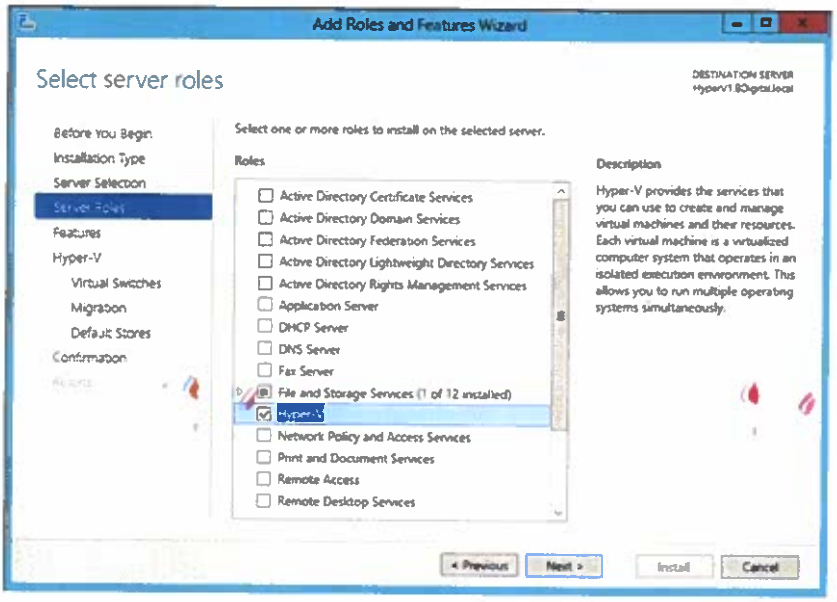

Figura 56 - Seleção da função *Hyper-V*

Seleciona-se o interface de rede que vai ficar associada ao a *Hyper-V*, figura 57.

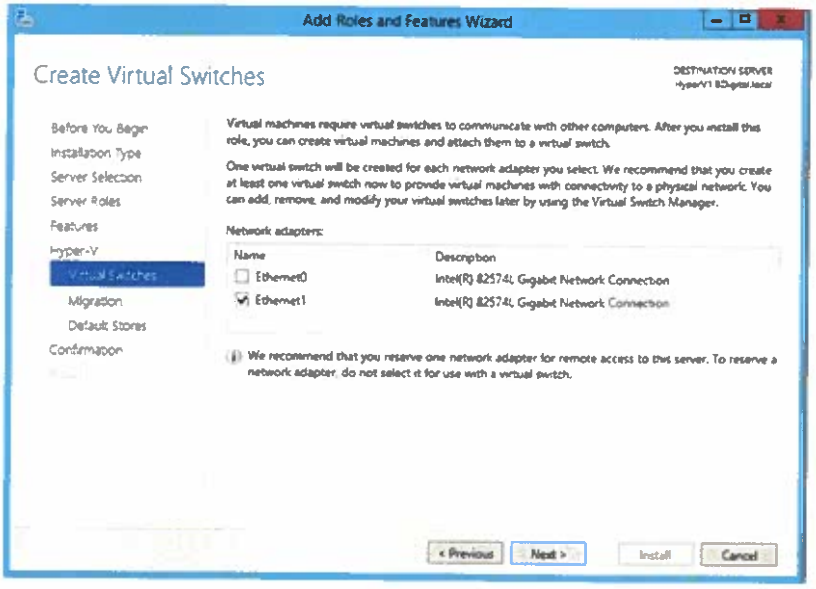

Figura 57 - Escolha do *interface* de rede

Mantém-se as definições por defeito para a localização dos discos e ficheiros de configuração das máquinas virtuais, figura 58.

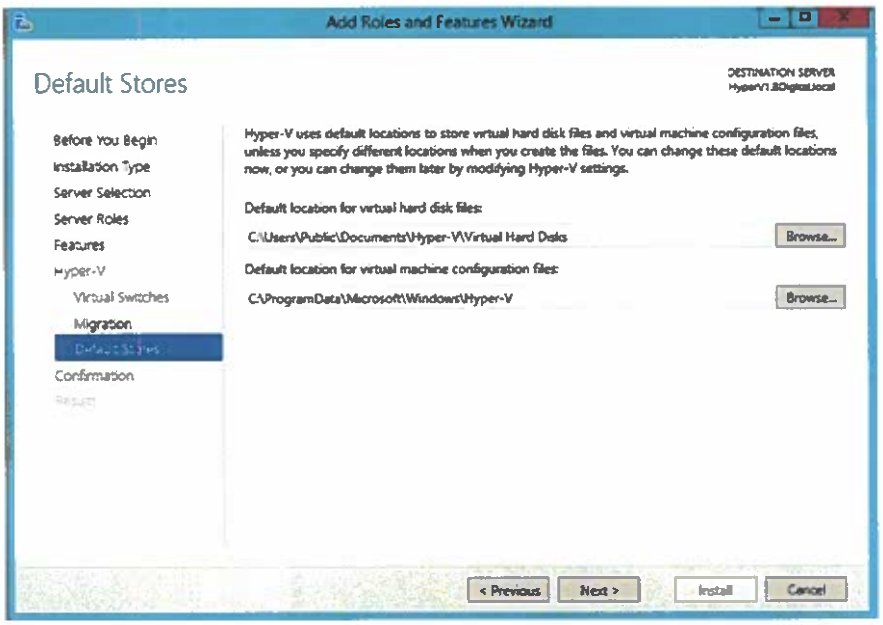

Figura 58 - Localização padrão

Processo de criação das maquinas virtuais no *HyperV-Manager* para execução da aplicação

Inicia-se a ferramenta *Hyper-V Manager*, conforme figura 59.

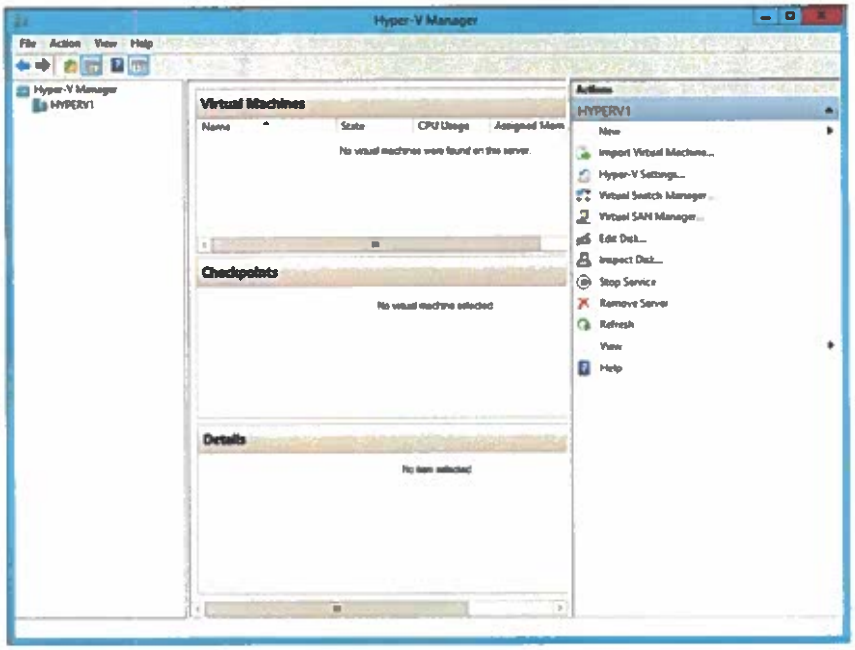

Figura 59 - Consola *Hyper-V Manager*

Definição do nome e localização da máquina virtual IT para administração aplicação, figura 60.

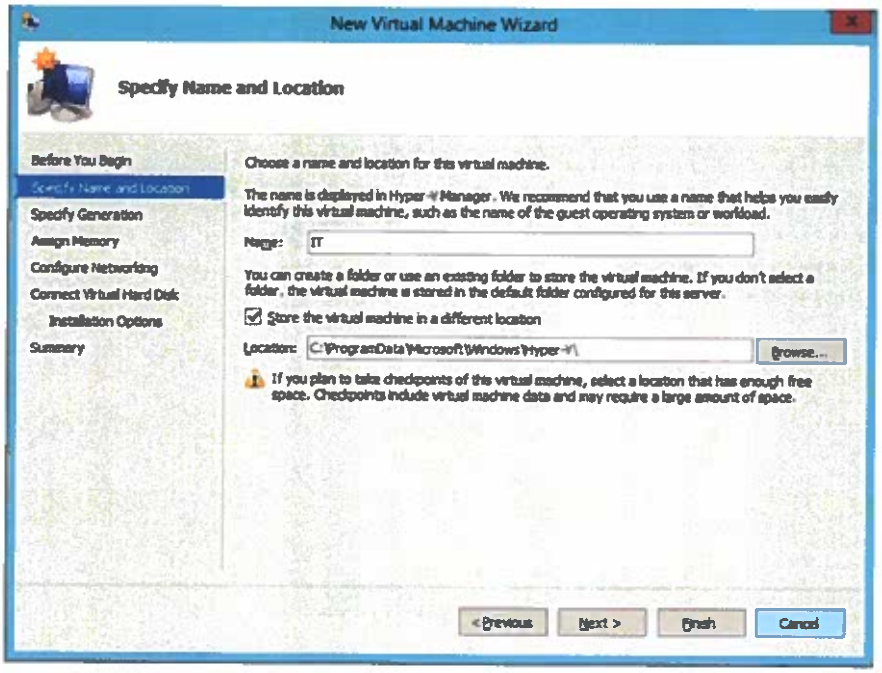

Figura 60 - Nome / Localização da máquina virtual cliente

Escolhemos a opção geração 1 para implementar a máquina virtual, figura 61.

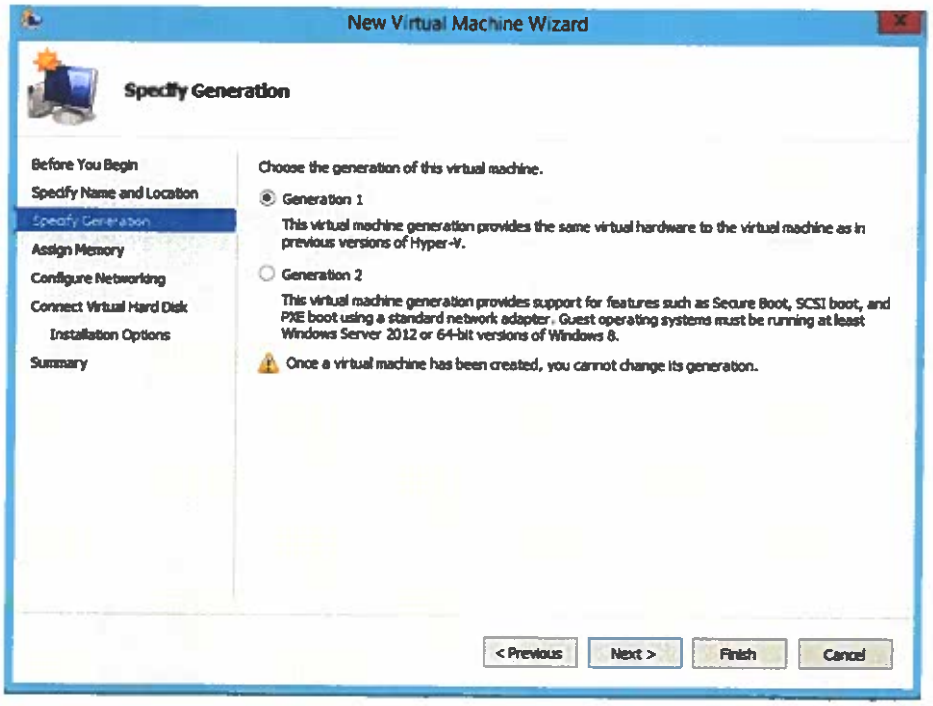

Figura 61 - Geração da máquina virtual

Aloca-se para memória 1024 MB, figura 62.

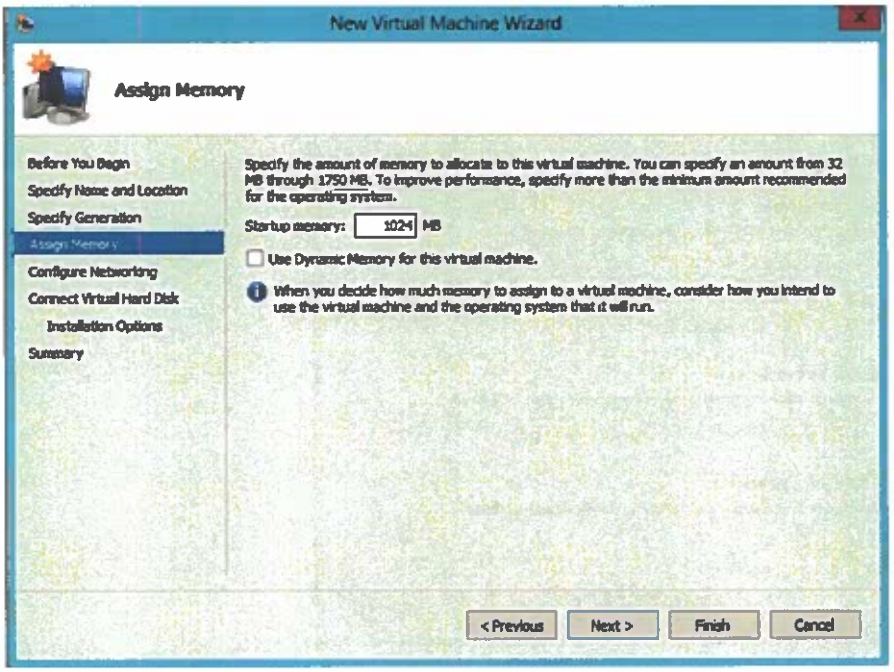

Figura 62 - Atribuição da memória

Seleciona-se o virtual switch para a conexão de rede, figura 63.

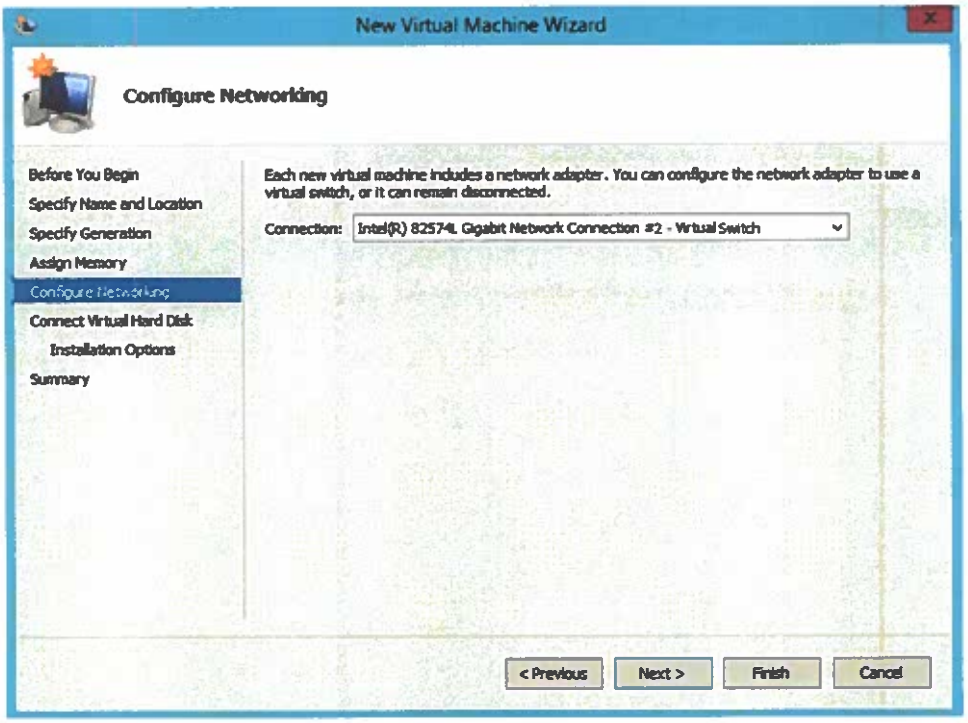

Figura 63 - Adaptador de rede

Escolhe-se a opção que permite adicionar posteriormente um disco virtual, figura 64.

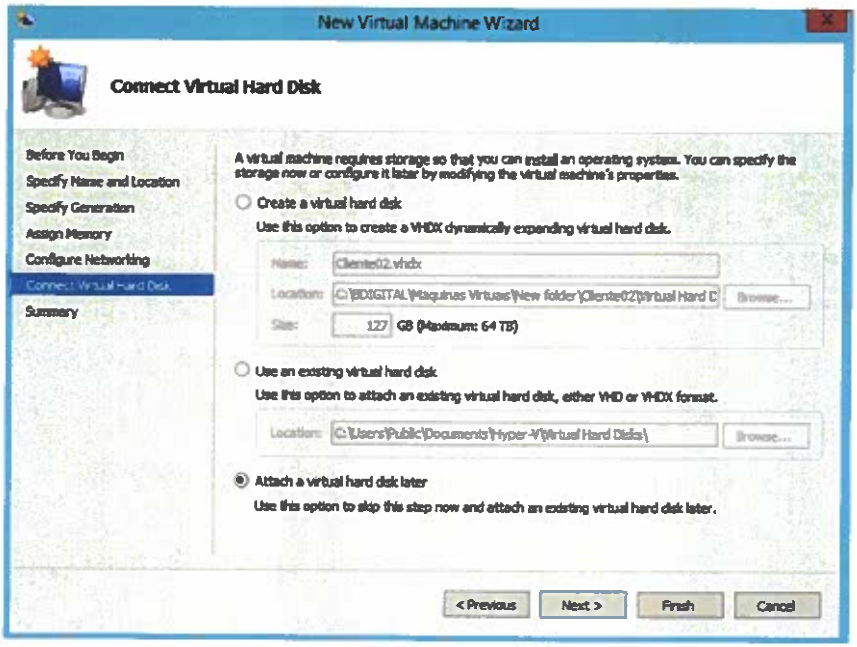

Figura 64 - Ligação para disco virtual

No painel de configuração da máquina virtual, escolhe-se o tipo de controladora IDE e o tipo de drive que se vai adicionar à controladora, figura 65.

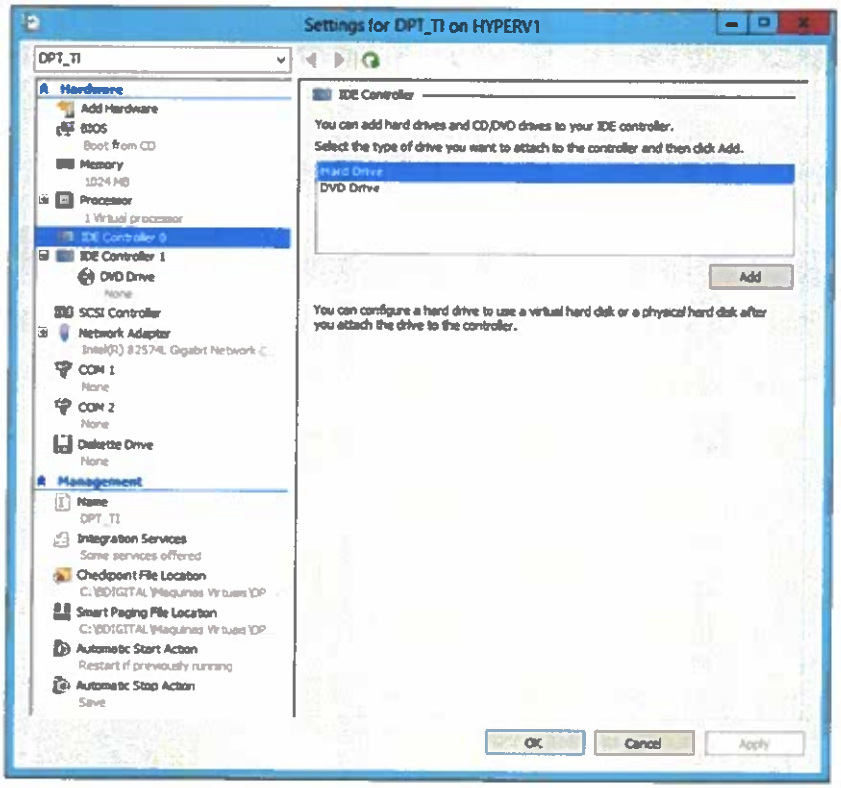

Figura 65 - Definições da máquina virtual

Escolhe-se o formato de disco *VHDX* como esta representado na figura 66.

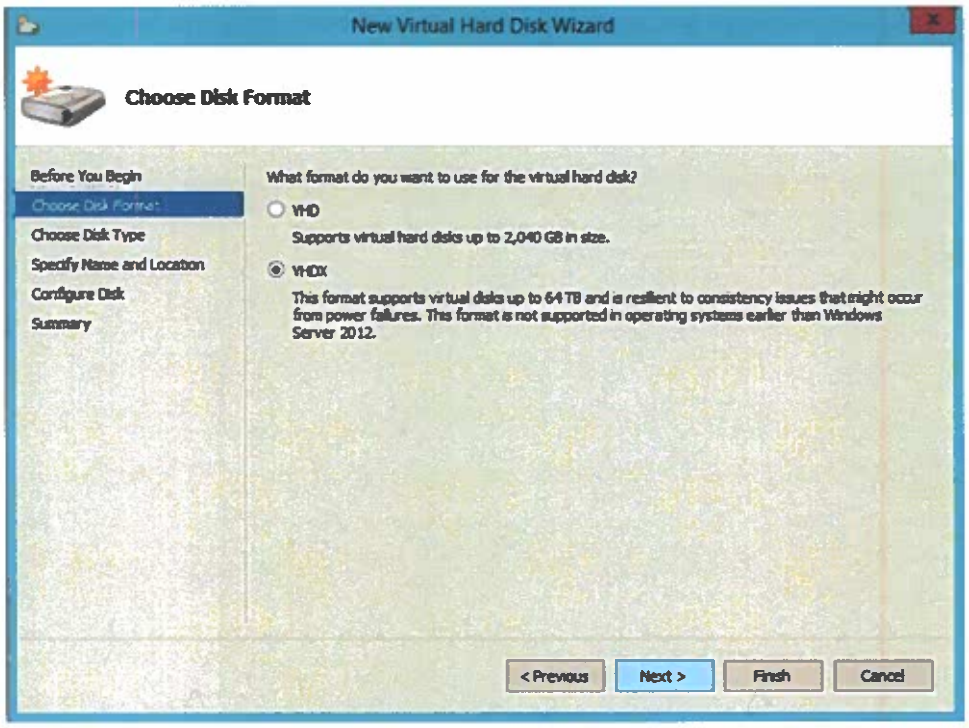

Figura 66 - Formato do disco virtual

Selecciona-se o tipo de disco pretendido, neste caso com expansão dinâmica, figura 67.

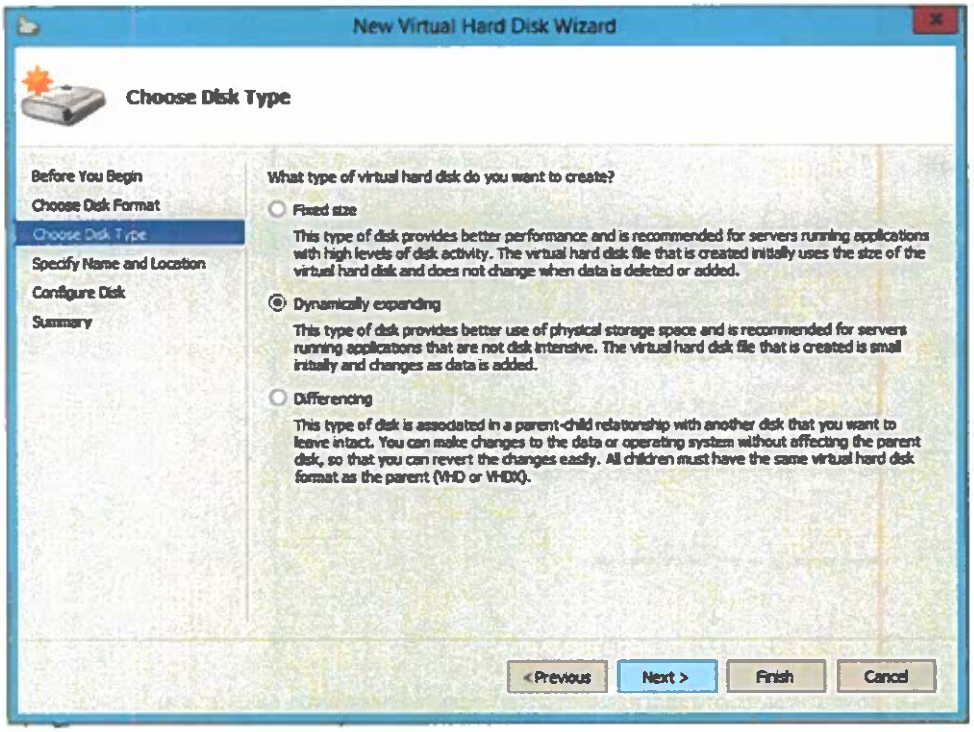

Figura 67 - Tipo de disco virtual

Escolher o nome e localização do disco virtual, figura 68.

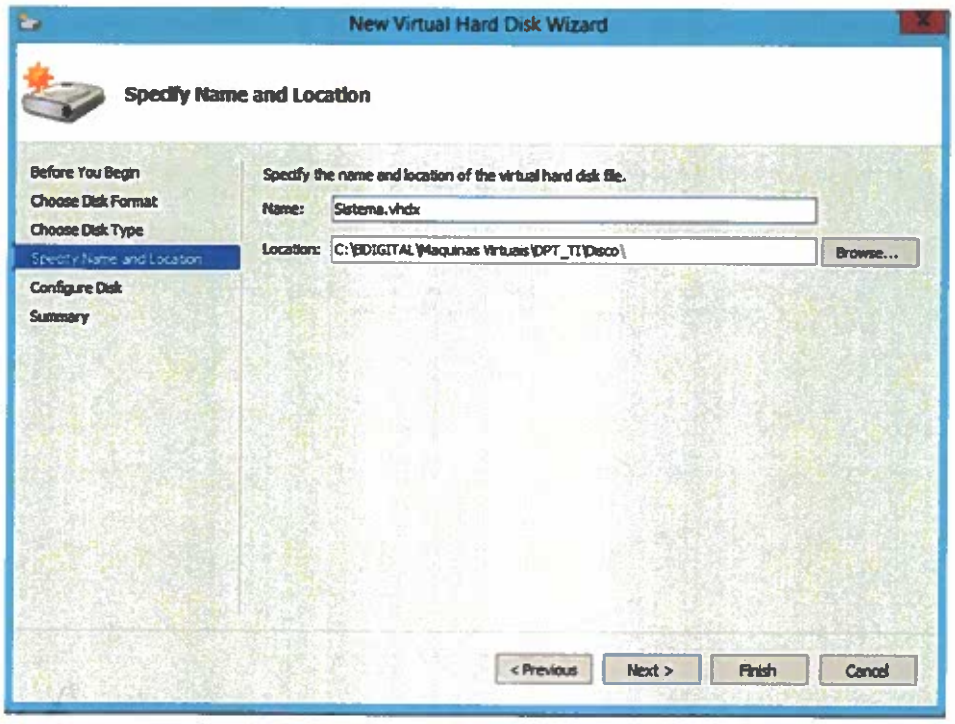

Figura 68 - Localização do disco virtual

Na figura 69 determina-se o tamanho do disco.

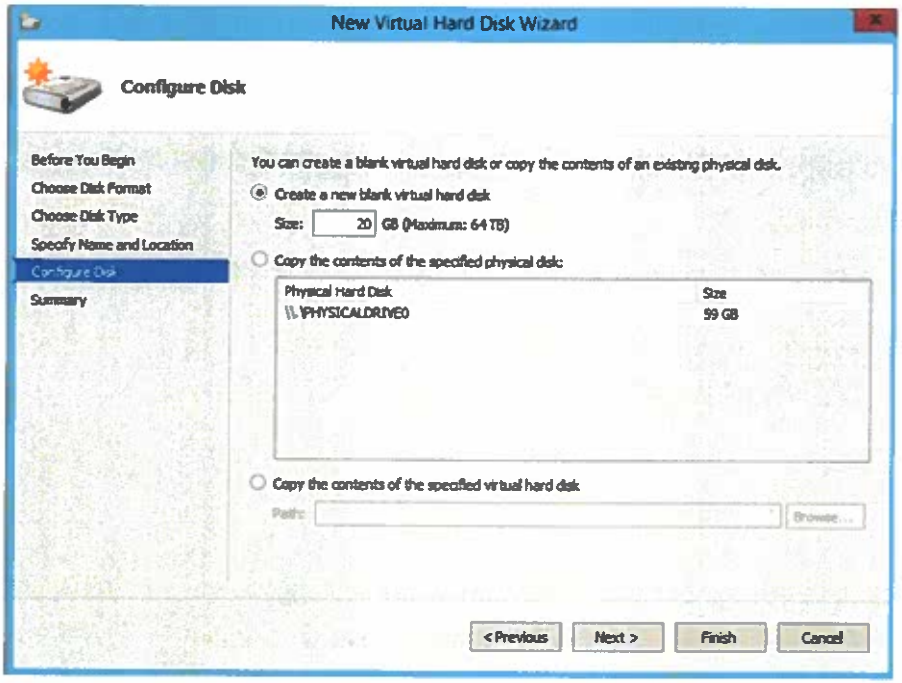

Figura 69 - Tamanho do disco virtual

Na figura 70 é apresentado um sumario com as configurações que foram pretendidas para o disco virtual.

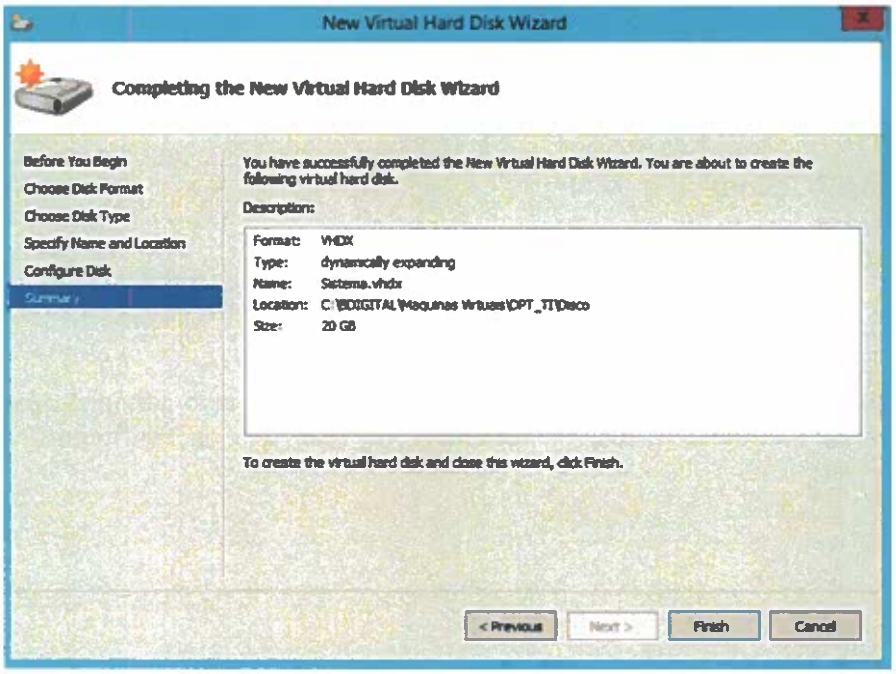

Figura 70 - Resumo da configuração do disco virtual

Inicia-se a instalaçao do sistema operativo designada para o cliente *Windows 8.1* pro e adiciona-se ao domínio seguindo os mesmos passos que anteriormente se efetuou nas maquinas virtuais BD1 e *Hyperv1*.

Configuração de *interfaces* de rede

Servidor DC:

Endereço de IP: 192.168.100.1

Mascara de sub-rede: 255.255.255.0

Gateway Padrão: 192.168.100.1

DNS: 192.168.100.1

Servidor BD1

Endereço de IP: 192.168.100.10

Mascara de sub-rede: 255.255.255.0

Gateway Padrão: 192.168.100.1

DNS: 192.168.100.1

Servidor *HyperV1*

Endereço de IP: 192.168.100.20

Mascara de sub-rede: 255.255.255.0

Gateway Padrão: 192.168.100.1

DNS: 192.168.100.1

As maquinas virtuais onde a aplicação vai ser acedida remotamente pelo administrador de TI e pelos clientes da biblioteca virtual os interfaces de rede têm as configurações automáticas.

Conclusão

A aplicação Biblioteca Digital, cujo o seu desenvolvimento, implementação e utilização foi descrita neste documento, apresenta uma forma consistente do trabalho cooperativo em rede utilizando as tecnologias de informação e seus recursos para disponibilizar e gerir os produtos de uma biblioteca ao seu consumidor. A aplicação permite armazenar e disponibilizar de forma detalhada todos os livros neles inseridos bem como os seus utilizadores, dados pessoais e registos de pesquisa. Pelo fato dos livros estarem em formato digital, a aplicação Biblioteca Digital, permite um acesso mais rápido, consizo e sem perca de informação. Sendo que o fator de manutenção e desgaste dos livros devido ao seu manuseamento tenha sido eliminado.

Conseguiu-se minimizar os elevados custos financeiros que uma estrutura destas requer ao nível do *hardware* recorrendo à virtualização. Todo o software utlizado que foi descrito neste projeto para suportar a aplicação, engloba as mais recentes tecnologias, ferramentas e as suas versões são bastante estáveis, dessa forma a aplicação pode ser executada de forma clara e segura.

Bibliografia

Alemán, D. (2006). *Biblioteca digital o virtual.*

Almeida, J. (2011). *Virtualiação de Servidores.*

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

Bloor, R., & Jozwiak, R. (s.d.). *Moving to the Cloud with Pervasive PSQL.* The Bloor Group, white paper.

Boehm, A., & Mead, G. (2010). *Murach's ADO.NET 4 Database Programming With C#2010 4th Edition.*

Boehm, A., & Mead, G. (2010). *Murach's ADO.NET 4 Database Programming With C#2010 4th Edition.*

Buyya, R., Vecchiola, C., & Thamarai, S. (2013). *Mastering Clou Computing Foundations and Aplications Programming.* Morgan Kaufmann - ELSEVIER.

Catteddu, D., & Giles , H. (2009). *European Network and Information Security Agency.* ENISA.

Codd, E. F. (June 1970). *Reprinted from Communications of the ACM, v.13 n.6, p.377-387.*

Cooter, M. (2007). The ultimate Guide to cloud Computing. *MagBook.*

Ferreira, N. (2004). *ADO.NET.* Instituto de Engenharia do Porto.

Ferreira, N. (2004). *ADO.NET.* Instituto de Engenharia do Porto.

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

Goldberg, R. P. (1974). *Survey of Virtual Machine research.* Honeywell Information Systems and Harvard University.

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

Lima, J. C. (2012). *O que é o SQL e qual a sua importância.*

Longeau, T. (s.d.). La Virtualisation des systèmes d'information. Alcantis.

Longeau, T. (s.d.). La Virtualisation des systèmes d'information. Alcantis.

Mann, A. (2006). *Virtualization 101: "Technologies, Benefits, and Challenges".* A White Paper.

Marshal, D. (2011). *Virtualization Report.* Obtido de InfoWorld: http://www.infoworld.com/article/2621502/server-virtualization/vmware-may-be-losing-dominance-in-server-virtualization.html

Meira, P. F. (1997). *Sistemas de Bancos de Dados.* Unimar - Universidade de Marília F.C.T.

Patrick, T. (October 2010). Published by Microsoft Press. In T. Patrick. Microsoft ADO.NET 4.0 Step by Step.

Patrick, T. (October 2010). *Published by Microsoft Press.* Microsoft ADO.NET 4.0 Step by Step.

R. Groff, J., & N. Weinberg, P. (1999). *SQL: The Complete Reference, Mcgraw-hill.*

Sosinsky, B. (2011). *Cloud Computing Bible.* Wiley Publishing, Inc.

Strachey, C. (1959). Time Sharing in Large Fast Computers. *Proc. Intl. Conf. on Information Processing*, (pp. B336-B341).

Tsuguio Okano, M., & Favero de Andrade, F. (2008). O IMPACTO DA VIRTUALIZAÇÃO NAS. *IV Congresso Nacional de Excelência em Gestão.* FIAP.

Turion, C. (2009). *Cloud Computing.* Brasport Livros e Multimídia Ltda.

Veras, M., & Kassick, R. (2011). *Administração de sistemas, Virtualização de servidores.* Escola Superior de Redes.

W. Van Den Bent, J., & Van Der Steeg, M. (2012). *"CLOUD-linguistics", "The Workbook Company".* EXIN Holding B.V.

ANEXOS

Anexo I - myQueriesTableAdapter.cs

```
1 using System;

2 using System.Collections.Generic;

3 using System.Linq;

4 using System.Text;

5

6 namespace istec.pg

7 {

8 /// <summary>

9 /// Classe de complemento à classe SearchesTableAdapters.QueriesTableAdapter

10 /// para poder alterar a ConnectionString dos Commands

11 /// </summary>

12 public class myQueriesTableAdapter : SearchesTableAdapters.QueriesTableAdapter

13 {

14 /// <summary>

15 /// Altera a ConnectionString dos Commands

16 /// </summary>
```

```
17 /// <param name="connectionString">ConnectionString a alterar</param>
18 public void ChangeConnection(string connectionString)
19 {
20 // Iterar os Commands do objeto para a alteração da ConnectionString
21 foreach (System.Data.IDbCommand command in this.CommandCollection)
22 command.Connection.ConnectionString = connectionString;
23 }
24 }
25 }
```

Anexo II - AdminView.xaml

```
1 using Microsoft.Win32;

2 using System;

3 using System.Collections.Generic;

4 using System.Configuration;

5 using System.IO;

6 using System.Linq;

7 using System.Text;

8 using System.Windows;

9 using System.Windows.Controls;

10 using System.Windows.Data;

11 using System.Windows.Documents;

12 using System.Windows.Input;

13 using System.Windows.Media;

14 using System.Windows.Media.Imaging;

15 using System.Windows.Shapes;

16
```

```
17 namespace istec.pg
18 {
19 /// <summary>
20 /// Interaction logic for AdminView.xaml
21 /// </summary>
22 public partial class AdminView : Window
23 {
24 /// <summary>
25 /// Variáveis privadas
26 /// </summary>
27 #region Privates
28 int current_book_id = -1;
29 int total_rows_found = 0;
30 bool loadingCombo = false;
31 private enmAdminMode _mode;
32 bool loading_window = true;
33 bool hasBookCover = false;
34 bool hasBookPDF = false;
```

```
35 #endregion
36
37 #region Enums & Properties
38 /// <summary>
39 /// Enum de Modo de Administração
40 /// </summary>
41 public enum enmAdminMode
42 {
43 Search, //Modo de pesquisa
44 Navigate, //Modo de navegação
45 EditBook, //Modo de edição do livro atual
46 NewBook //Modo de criação de um livro
47 }
48 /// <summary>
49 /// Modo de Administração
50 /// </summary>
51 public enmAdminMode mode
52 {
```

```
53 get
54 {
55 return _mode;
56 }
57 set
58 {
59 _mode = value;
60 }
61 }
62 #endregion
63
64 #region Window Methods
65 /// <summary>
66 /// Construtor da janela
67 /// </summary>
68 public AdminView()
69 {
70 //Inicialização dos controlos
```

```
71 InitializeComponent();
72
73 //Se o user que iniciou sessão for um administrador
74 if (Utils.UserType == 1)
75 lblTitle.Content = "Biblioteca Digital - Administração";
76
77 //Se o user que iniciou sessão for um utilizador normal
78 if (Utils.UserType == 2)
79 lblTitle.Content = "Biblioteca Digital";
80 }
81 /// <summary>
82 /// Carregamento da janela
83 /// Inicialização de propriedades de vários controlos
84 /// </summary>
85 /// <param name="sender"></param>
86 /// <param name="e"></param>
87 private void Window_Loaded(object sender, RoutedEventArgs e)
88 {
```

```
89 txtBookDescription.HorizontalScrollBarVisibility = ScrollBarVisibility.Visible;

90 txtBookDescription.Document.PageWidth = 1000;

91

92 txtUsername.Text = Utils.CurrentUsername;

93 txtUserCategory.Text = Utils.GetCurrentUserTypeDescription();

94 txtLastLogin.Text = string.Format("{0:yyyy-MM-dd HH:mm:ss}", Utils.LastLogin);

95

96 //Imagens 'Default' da pré-visualização do livro e da photo de utilizador

97 imgBookCover.Source = new BitmapImage(new Uri("/Images/NotAvailable.png",
UriKind.Relative))

;

98 imgUserPhoto.Source = new BitmapImage(new Uri("/Images/no_person_available.png",
UriKind.

Relative));

99

100 //Carregamento das editoras

101 LoadPublishers();

102 //Carregamento das pesquisas efetuadas anteriormente

103 LoadSearches();
```

```
104

105 //Variável para detetar que o carregamento terminou

106 loading_window = false;

107

108 //Iniciar como modo de pesquisa

109 mode = enmAdminMode.Search;

110 //Habilitar/desabilitar controlos baseado nas variáveis 'mode' e 'Utils.UserType'

111 SetEnvironment();

112

113 //Obter a photo do utilizador com sessão iniciada

114 imgUserPhoto.Source = Utils.GetUserPhoto(Utils.CurrentUserID);

115 }

116 #endregion

117

118 #region Loadings...

119 /// <summary>

120 /// Carregamento das editoras na combo

121 /// </summary>
```

```
122 private void LoadPublishers()
123 {
124 //Ligação à base de dados através de um TableAdapter
125 PublishersTableAdapters.TPublishersTableAdapter tpta = new PublishersTableAdapters.
TPublishersTableAdapter();
126                                    tpta.Connection.ConnectionString                    =
ConfigurationManager.ConnectionStrings["BooksDatabase.
Properties.Settings.BooksLibraryConnectionString"].ConnectionString;
127 //Carregamento da tabela
128 Publishers.TPublishersDataTable tpdt = tpta.GetData_TPublishers(null, null);
129
130 //Limpar a datagrid
131 cmbPublisher.Items.Clear();
132
133 //Carregamento da combo
134 cmbPublisher.DisplayMemberPath = "Key";
135 cmbPublisher.SelectedValuePath = "Value";
136 foreach (Publishers.TPublishersRow row in tpdt.Rows)
```

```
137 cmbPublisher.Items.Add(new KeyValuePair<string, int>(row.description, row.id));
138
139 //Se houver pelo menos 1 registo, selecioná-lo
140 if (cmbPublisher.Items.Count > 0) cmbPublisher.SelectedIndex = 0;
141 }
142 /// <summary>
143 /// Carregamento das últimas pesquisas na lista
144 /// </summary>
145 private void LoadSearches()
146 {
147 //Ligação à base de dados através de um TableAdapter
148 SearchesTableAdapters.TSearchesTableAdapter tpta = new SearchesTableAdapters.
TSearchesTableAdapter();
149 tpta.Connection.ConnectionString =
ConfigurationManager.ConnectionStrings["BooksDatabase.
Properties.Settings.BooksLibraryConnectionString"].ConnectionString;
150 //Carregamento da tabela
151 Searches.TSearchesDataTable tpdt = tpta.GetDataBy_UserID(Utils.CurrentUserID, null,
null);
```

```
152

153 //Ligação à base de dados através de um TableAdapter

154 SearchesTableAdapters.UserSearches_GetTableAdapter usgta = new
SearchesTableAdapters.

UserSearches_GetTableAdapter();

155 usgta.Connection.ConnectionString =
ConfigurationManager.ConnectionStrings["BooksDatabase.

Properties.Settings.BooksLibraryConnectionString"].ConnectionString;

156 //Carregamento da tabela

157 Searches.UserSearches_GetDataTable usgdt = usgta.GetData_GetUsersSearches(Utils.

CurrentUserID);

158

159 //Carregamento da combo

160 lstLastSearches.Items.Clear();

161 lstLastSearches.DisplayMemberPath = "Key";

162 lstLastSearches.SelectedValuePath = "Value";

163 foreach (Searches.UserSearches_GetRow row in usgdt.Rows)

164 lstLastSearches.Items.Add(new KeyValuePair<string, int>(string.Format("[{1:yyyy-
MM-dd HH
```

```
:mm:ss}] - {0}", row.search_text, row.date), row.id));
165 }
166 /// <summary>
167 /// Carregamento do livro selecionado
168 /// </summary>
169 private void LoadCurrentSelectedBook()
170 {
171 //Se não houver um livro selecionado
172 if (cmbBooks.SelectedItem == null)
173 {
174 //Limpar os controlos
175 txtReference.Text = string.Empty;
176 txtTitle.Text = string.Empty;
177 txtAuthor.Text = string.Empty;
178 cmbPublisher.SelectedIndex = -1;
179 txtBookDescription.Document.Blocks.Clear();
180
181 //Carregar imagem de livro não disponível
```

```
182 byte[] picbyte_empty = Utils.GetFileBytes(@"~/Images/NotAvailable.png", "Image
Files (*.

bmp; *.jpg; *.png; *.gif)|*.bmp; *.jpg; *.png; *.gif|Todos os ficheiros (*.*)|*.*", "Escolher
uma

imagem para o livro");

183 imgBookCover.Source = Utils.SetImageFromByteArray(picbyte_empty);

184

185 //Não existe capa de livro

186 hasBookCover = false;

187 //Não existe PDF do livro

188 hasBookPDF = false;

189

190 //Sair da função

191 return;

192 }

193

194 //Obter o ID do livro selecionado

195 int book_id = ((KeyValuePair<string, int>)cmbBooks.SelectedItem).Value;

196
```

```
197 //Ligação à base de dados através de um TableAdapter

198 BooksTableAdapters.TBooksTableAdapter tbta = new
BooksTableAdapters.TBooksTableAdapter();

199 tbta.Connection.ConnectionString =
ConfigurationManager.ConnectionStrings["BooksDatabase.

Properties.Settings.BooksLibraryConnectionString"].ConnectionString;

200 //Carregamento da tabela

201 Books.TBooksDataTable tbdt = tbta.GetData_TBooks(book_id, null, null, null, null,
null);

202

203 //Se encontrou o registo

204 if (tbdt != null && tbdt.Rows.Count > 0)

205 {

206 Books.TBooksRow row = ((Books.TBooksRow)tbdt.Rows[0]);

207 if (row != null)

208 {

209 //Guardar o modo para mais tarde o voltar a obter

210 enmAdminMode old_mode = mode;

211
```

```
212 //Colocar os valores nos respetivos controlos
213 txtReference.Text = row.reference;
214 txtTitle.Text = row.title;
215 txtAuthor.Text = row.author;
216 Utils.SelectComboByID(cmbPublisher, row.publisher);
217 txtBookDescription.Document.Blocks.Clear();
218 txtBookDescription.Selection.Text = row.book_notes;
219
220 //Variável que mostra se tem ou não um PDF
221 hasBookPDF = (row.IsbookNull() == false);
222
223 //Se existe uma pré-visualização do livro
224 if (row.Isbook_thumbnailNull() == false)
225 {
226 //Abrir a imagem
227 byte[] picbyte = row.book_thumbnail;
228 imgBookCover.Source = Utils.SetImageFromByteArray(row.book_thumbnail);
229 //Existe capa de livro
```

```
230 hasBookCover = true;
231 }
232 else
233 {
234 //Abrir uma imagem de capa não disponível
235 byte[] picbyte_empty = Utils.GetFileBytes(@"~/Images/NotAvailable.png", "Image
Files (*.bmp; *.jpg; *.png; *.gif)|*.bmp; *.jpg; *.png; *.gif|Todos os ficheiros (*.*)|*.*",
"Escolher uma imagem para o livro");
236 imgBookCover.Source = Utils.SetImageFromByteArray(picbyte_empty);
237 //Não existe capa de livro
238 hasBookCover = false;
239 }
240
241 //Recolocar o modo guardado
242 mode = old_mode;
243 }
244 }
245 }
```

```
246 #endregion

247

248 #region Buttons

249 private void btnSearch_Click(object sender, RoutedEventArgs e)

250 {

251 //Modo de pesquisa

252 mode = enmAdminMode.Search;

253

254 bool search_exists = false;

255 bool user_has_searched = false;

256 int id_search = -1;

257

258 //Verificar se a procura já existe na base de dados

259 //Ligação à base de dados através de um TableAdapter

260 SearchesTableAdapters.TSearchesTableAdapter tsta = new SearchesTableAdapters.

TSearchesTableAdapter();

261 tsta.Connection.ConnectionString =

ConfigurationManager.ConnectionStrings["BooksDatabase.
```

```
Properties.Settings.BooksLibraryConnectionString"].ConnectionString;
262 //Carregamento da tabela
263 Searches.TSearchesDataTable tsdt = tsta.GetData_TSearches(null, txtSearch.Text);
264 search_exists = (tsdt != null && tsdt.Rows.Count >= 1);
265
266 //Se a pesquisa existe
267 if (search_exists)
268 {
269 Searches.TSearchesRow row = ((Searches.TSearchesRow)tsdt.Rows[0]);
270 id_search = row.id;
271
272 row = null;
273 tsdt = null;
274 tsta = null;
275
276 //Verificar se a procura já foi feita pelo utilizador
277 //Ligação à base de dados através de um TableAdapter
```

```
278 SearchesTableAdapters.TUsersSearchesTableAdapter tusta = new
SearchesTableAdapters.
TUsersSearchesTableAdapter();
279 tusta.Connection.ConnectionString = ConfigurationManager.ConnectionStrings[
"BooksDatabase.Properties.Settings.BooksLibraryConnectionString"].ConnectionString;
280 //Carregamento da tabela
281 Searches.TUsersSearchesDataTable tusdt = tusta.GetData_TUsersSearches(Utils.
CurrentUserID, id_search, null);
282 user_has_searched = (tusdt != null && tusdt.Rows.Count >= 1);
283
284 //Limpeza de memória
285 tusdt = null;
286 tusta = null;
287 }
288 else
289 {
290 //Inserção da pesquisa na base de dados
291 Searches.TSearchesRow row = tsdt.NewTSearchesRow();
```

```
292 row.search_text = txtSearch.Text;

293 tsdt.Rows.Add(row);

294 int ret_aux = tsta.Update(tsdt);

295 if (ret_aux > 0)

296 {

297 id_search = row.id;

298 search_exists = true;

299 }

300 }

301

302 //Se o utilizador já pesquisou por esta pesquisa

303 if (user_has_searched == false)

304 {

305 //Ligação à base de dados através de um TableAdapter

306 SearchesTableAdapters.TUsersSearchesTableAdapter tusta_aux = new
SearchesTableAdapters.

TUsersSearchesTableAdapter();

307 tusta_aux.Connection.ConnectionString = ConfigurationManager.ConnectionStrings[
```

```
"BooksDatabase.Properties.Settings.BooksLibraryConnectionString"].ConnectionString;

308 //Carregamento da tabela

309 Searches.TUsersSearchesDataTable tusdt_aux =
tusta_aux.GetData_TUsersSearches(Utils.

CurrentUserID, id_search, null);

310

311 //Se encontrou a pesquisa

312 if (tusdt_aux != null && tusdt_aux.Rows.Count == 0)

313 {

314 //Atualizar a data da pesquisa efetuada

315 int ret2 = Utils.UserSearch_UpdateDate(Utils.CurrentUserID, id_search,
DateTime.Now)

;

316 if (ret2 >= 1)

317 LoadSearches();

318 else

319 {

320 MessageBox.Show("A pesquisa não foi atualizada. Houve um erro", "Atualizar data

da pesquisa", MessageBoxButton.OK, MessageBoxImage.Exclamation);
```

```
321 return;
322 }
323 }
324 }
325
326 //Carregamento dos livros baseados na pesquisa efetuada
327 FillComboOfBooksFound();
328
329 //Ligação à base de dados através de um TableAdapter
330 myQueriesTableAdapter mqta = new myQueriesTableAdapter();
331 //Alteração da ConnectionString com base no que está configurado no ficheiro .config
332
mqta.ChangeConnection(ConfigurationManager.ConnectionStrings["BooksDatabase.Properti
es.
Settings.BooksLibraryConnectionString"].ConnectionString);
333
334 //Atualizar a data da pesquisa efetuada
335 int retx = mqta.UsersSearches_UpdateSearchDate(Utils.CurrentUserID, txtSearch.Text,
DateTime
```

```
.Now);
336 if (retx >= 1)
337 LoadSearches();
338 else
339 {
340 MessageBox.Show("A pesquisa não foi atualizada. Houve um erro", "Atualizar data da
pesquisa", MessageBoxButton.OK, MessageBoxImage.Exclamation);
341 return;
342 }
343
344 //Habilitar/desabilitar controlos baseado nas variáveis 'mode' e 'Utils.UserType'
345 SetEnvironment();
346 }
347 private void btnNewBook_Click(object sender, RoutedEventArgs e)
348 {
349 txtReference.Text = string.Empty;
350 txtTitle.Text = string.Empty;
351 txtAuthor.Text = string.Empty;
```

```
352 cmbBooks.SelectedIndex = -1;
353 txtBookDescription.Document.Blocks.Clear(); //?!?
354
355 mode = enmAdminMode.NewBook;
356 //Habilitar/desabilitar controlos baseado nas variáveis 'mode' e 'Utils.UserType'
357 SetEnvironment();
358 }
359 private void btnCancel_Click(object sender, RoutedEventArgs e)
360 {
361 if (mode == enmAdminMode.NewBook)
362 current_book_id = 0;
363 cmbBooks.SelectedIndex = current_book_id;
364 loading_window = true;
365 LoadCurrentSelectedBook();
366 loading_window = false;
367 mode = enmAdminMode.Search;
368 //Habilitar/desabilitar controlos baseado nas variáveis 'mode' e 'Utils.UserType'
369 SetEnvironment();
```

```
370 }
371 private void btnSaveBook_Click(object sender, RoutedEventArgs e)
372 {
373 try
374 {
375 //Verificar se os campos estão preenchidos
376 if (txtReference.Text.Trim() == string.Empty ||
377 txtTitle.Text.Trim() == string.Empty ||
378 txtAuthor.Text.Trim() == string.Empty ||
379 cmbPublisher.SelectedIndex < 0 ||
380 IsRTBEmpty(txtBookDescription) == false)
381 {
382 MessageBox.Show("Tem de preencher os campos para criar um livro novo.", "Gravar
Livro", MessageBoxButton.OK, MessageBoxImage.Exclamation);
383 return;
384 }
385
386 //Validar tamanho do campo do resumo do livro
```

```
387 txtBookDescription.SelectAll();

388 if (txtBookDescription.Selection.Text.Length > 2000)

389 {

390 MessageBox.Show("O resumo do livro não pode conter mais do que 2000 caracteres.\r\

nTotal: " + txtBookDescription.Selection.Text.Length.ToString() + " caracteres
encontrados.\r\n\r\

nReduza o texto por forma a poder gravar", "Gravar Livro", MessageBoxButton.OK,
MessageBoxImage.

Exclamation);

391 return;

392 }

393

394 //Se o modo é a criação de um livro novo

395 if (mode == enmAdminMode.NewBook)

396 {

397 //Ligação à base de dados através de um TableAdapter

398 BooksTableAdapters.TBooksTableAdapter tbta = new BooksTableAdapters.

TBooksTableAdapter();

399 tbta.Connection.ConnectionString = ConfigurationManager.ConnectionStrings[
```

```
"BooksDatabase.Properties.Settings.BooksLibraryConnectionString"].ConnectionString;
400 //Carregamento da tabela
401 Books.TBooksDataTable tbdt = tbta.GetData_TBooks(null, null, null, null, null, null)
;
402
403 int ret = -1;
404 if (tbdt != null)
405 {
406 string reference = null;
407 reference = txtReference.Text;
408
409 //Ligação à base de dados através de um TableAdapter
410 BooksTableAdapters.TBooksTableAdapter tbta_2 = new BooksTableAdapters.
TBooksTableAdapter();
411 tbta_2.Connection.ConnectionString = ConfigurationManager.ConnectionStrings[
"BooksDatabase.Properties.Settings.BooksLibraryConnectionString"].ConnectionString;
412 //Carregamento da tabela
413 Books.TBooksDataTable tbdt_2 = tbta_2.GetData_TBooks(null, reference, null, null
```

```
, null, null);
414 if (tbdt_2 != null && tbdt_2.Rows.Count >= 1)
415 {
416 MessageBox.Show("Erro a adicionar o Livro.\r\nA referência '" + txtReference
.Text + "' já existe.", "Erro", MessageBoxButton.OK, MessageBoxImage.Exclamation);
417 return;
418 }
419
420 Books.TBooksRow rowToSave = null;
421 //Se fôr no modo de novo livro adicionar uma nova 'row'
422 if (mode == enmAdminMode.NewBook)
423 {
424 rowToSave = tbdt.NewTBooksRow();
425 }
426
427 //Atribuir os valores aos campos da 'row'
428 rowToSave.reference = txtReference.Text;
429 rowToSave.title = txtTitle.Text;
```

```
430 rowToSave.author = txtAuthor.Text;

431 int publisher_id = ((KeyValuePair<string, int>)cmbPublisher.SelectedItem).Value;

432 rowToSave.publisher = publisher_id;

433 txtBookDescription.SelectAll();

434 rowToSave.book_notes = txtBookDescription.Selection.Text;

435

436 //Se fôr no modo de novo livro adicionar a 'row' à tabela

437 if (mode == enmAdminMode.NewBook)

438 tbdt.Rows.Add(rowToSave);

439

440 try

441 {

442 //Criar/Atualizar o registo

443 ret = tbta.Update(rowToSave);

444 if (ret == 1)

445 {

446 //Adicionado com sucesso!

447 }
```

```
448 }
449 catch (Exception ex)
450 {
451 MessageBox.Show("Erro: " + ex.Message, "Erro", MessageBoxButton.OK,
MessageBoxImage.Error);
452 }
453 }
454 }
455
456 //Se o modo é a alteração de um livro existente
457 if (mode == enmAdminMode.EditBook)
458 {
459 //Obter o ID do livro selecionado
460 int book_id = ((KeyValuePair<string, int>)cmbBooks.SelectedItem).Value;
461
462 //Ligação à base de dados através de um TableAdapter
463 BooksTableAdapters.TBooksTableAdapter tbta = new BooksTableAdapters.
TBooksTableAdapter();
```

```
464 tbta.Connection.ConnectionString = ConfigurationManager.ConnectionStrings[
"BooksDatabase.Properties.Settings.BooksLibraryConnectionString"].ConnectionString;
465 //Carregamento da tabela
466 Books.TBooksDataTable tbdt = tbta.GetData_TBooks(book_id, null, null, null, null,
null);
467
468 int ret = -1;
469 if (tbdt != null)
470 {
471 //Ligação à base de dados através de um TableAdapter
472 BooksTableAdapters.TBooksTableAdapter tbta_2 = new BooksTableAdapters.
TBooksTableAdapter();
473 tbta_2.Connection.ConnectionString = ConfigurationManager.ConnectionStrings[
"BooksDatabase.Properties.Settings.BooksLibraryConnectionString"].ConnectionString;
474 //Carregamento da tabela
475 Books.TBooksDataTable tbdt_2 = tbta_2.GetData_TBooks(null, txtReference.Text,
null, null, null, null);
476
```

```
477 //Se encontrou o registo
478 if (tbdt_2 != null && tbdt_2.Rows.Count >= 1)
479 {
480 Books.TBooksRow rowToEdit = ((Books.TBooksRow)tbdt_2.Rows[0]);
481
482 //Verificar se a referência foi encontrada noutro livro
483 if (rowToEdit.reference == txtReference.Text && rowToEdit.id != book_id)
484 {
485 MessageBox.Show("Erro a alterar o Livro.\r\nA referência '" +
txtReference.Text + "' já existe.", "Erro", MessageBoxButton.OK,
MessageBoxImage.Exclamation);
486 return;
487 }
488 }
489
490 Books.TBooksRow rowToSave = null;
491
492 //Se o modo é a criação de um livro novo
```

```
493 if (mode == enmAdminMode.NewBook)
494 {
495 //Adicionar uma nova linha à tabela
496 rowToSave = tbdt.NewTBooksRow();
497 }
498 //Se o modo é a alteração de um livro existente
499 if (mode == enmAdminMode.EditBook)
500 {
501 //Obter o registo encontrado
502 rowToSave = tbdt.FindByid(book_id);
503 }
504
505 //Colocar os valores nos respetivos campos do registo
506 rowToSave.reference = txtReference.Text;
507 rowToSave.title = txtTitle.Text;
508 rowToSave.author = txtAuthor.Text;
509 int publisher_id = ((KeyValuePair<string, int>)cmbPublisher.SelectedItem).Value;
510 rowToSave.publisher = publisher_id;
```

```
511 txtBookDescription.SelectAll();

512 rowToSave.book_notes = txtBookDescription.Selection.Text;

513

514 //Se o modo é a criação de um livro novo

515 if (mode == enmAdminMode.NewBook)

516 {

517 //Adicionar a 'row' à tabela

518 tbdt.Rows.Add(rowToSave);

519 }

520

521 try

522 {

523 //Atualizar a base de dados

524 ret = tbta.Update(rowToSave);

525 if (ret == 1)

526 {

527 //Atualizado com sucesso

528 }
```

```
529 }
530 catch (Exception ex)
531 {
532 MessageBox.Show("Erro: " + ex.Message, "Erro", MessageBoxButton.OK,
MessageBoxImage.Error);
533 }
534 }
535 }
536
537 //Limpar os controlos
538 txtReference.Text = string.Empty;
539 txtTitle.Text = string.Empty;
540 txtAuthor.Text = string.Empty;
541 cmbPublisher.SelectedIndex = 0;
542 txtBookDescription.Document.Blocks.Clear();
543
544 if (mode == enmAdminMode.NewBook)
545 current_book_id = 0;
```

```
546 cmbBooks.SelectedIndex = current_book_id;

547

548 //Voltar a carregar o livro selecionado

549 loading_window = true;

550 LoadCurrentSelectedBook();

551 loading_window = false;

552

553 //Modo de pesquisa

554 mode = enmAdminMode.Search;

555 //Habilitar/desabilitar controlos baseado nas variáveis 'mode' e 'Utils.UserType'

556 SetEnvironment();

557 }

558 catch (Exception ex)

559 {

560 MessageBox.Show("Erro: " + ex.Message, "Erro", MessageBoxButton.OK,
MessageBoxImage.

Error);

561 return;
```

```
562 }
563
564 //Carregamento dos livros baseados na pesquisa efetuada
565 FillComboOfBooksFound();
566 }
567 private void btnLogoff_Click(object sender, RoutedEventArgs e)
568 {
569 //Voltar a abrir a janela de Início de Sessão
570 Login newLoginWindow = new Login();
571 newLoginWindow.Show();
572
573 //Fechar a janela principal
574 this.Close();
575 }
576 private void btnPrevious_Click(object sender, RoutedEventArgs e)
577 {
578 //Modo de navegação
579 mode = enmAdminMode.Navigate;
```

```
580

581 //Navegar para trás

582 current_book_id--;

583 loadingCombo = true;

584 cmbBooks.SelectedIndex = current_book_id;

585

586 //Habilitar/desabilitar controlos baseado nas variáveis 'mode' e 'Utils.UserType'

587 SetEnvironment();

588 }

589 private void btnNext_Click(object sender, RoutedEventArgs e)

590 {

591 //Modo de navegação

592 mode = enmAdminMode.Navigate;

593

594 //Navegar para a frente

595 current_book_id++;

596 loadingCombo = true;

597 cmbBooks.SelectedIndex = current_book_id;
```

```
598 //Habilitar/desabilitar controlos baseado nas variáveis 'mode' e 'Utils.UserType'
599 SetEnvironment();
600 }
601 private void btnUpload_Click(object sender, RoutedEventArgs e)
602 {
603 //Obter os bytes de uma imagem
604 byte[] picbyte = Utils.GetFileBytes(null, "Image Files (*.bmp; *.jpg; *.png; *.gif)|*.bmp;
*
.jpg; *.png; *.gif|Todos os ficheiros (*.*)|*.*", "Escolher uma imagem para o livro");
605
606 if (picbyte != null)
607 {
608 //Obter o ID do livro selecionado
609 int book_id = ((KeyValuePair<string, int>)cmbBooks.SelectedItem).Value;
610
611 //Ligação à base de dados através de um TableAdapter
612 BooksTableAdapters.TBooksTableAdapter tbta = new
BooksTableAdapters.TBooksTableAdapter()
;
```

```
613                 tbta.Connection.ConnectionString =
ConfigurationManager.ConnectionStrings["BooksDatabase
.Properties.Settings.BooksLibraryConnectionString"].ConnectionString;
614 //Carregamento da tabela
615 Books.TBooksDataTable tbdt = tbta.GetData_TBooks(book_id, null, null, null, null,
null);
616
617 if (tbdt != null && tbdt.Rows.Count > 0)
618 {
619 Books.TBooksRow row = ((Books.TBooksRow)tbdt.Rows[0]);
620
621 //Se o registo foi encontrado
622 if (row != null)
623 {
624 //Atribuir a imagem ao campo
625 row.book_thumbnail = picbyte;
626
627 //Atualizar a pré-visualização do livro
628 int ret = tbta.Update(row);
```

```
629 if (ret == 1)
630 {
631 var Img = new BitmapImage();
632 Img.BeginInit();
633 Img.StreamSource = new System.IO.MemoryStream(picbyte);
634 Img.EndInit();
635
636 //Colocar a imagem no controlo
637 imgBookCover.Source = Img;
638
639 //Indicar que o livro tem uma pré-visualização
640 hasBookCover = true;
641 }
642 }
643 }
644 }
645
646 //Libertar memória
```

```
647 picbyte = null;

648 }

649 private void btnManageUsers_Click(object sender, RoutedEventArgs e)

650 {

651 //Abrir a janela de manutenção de utilizadores

652 ManageUsers manUsers = new ManageUsers();

653 manUsers.Owner = this;

654 manUsers.ShowDialog();

655

656 //Obter a foto do utilizador, caso o mesmo a tenha alterado na manutenção de utilizadores

657 imgUserPhoto.Source = Utils.GetUserPhoto(Utils.CurrentUserID);

658 }

659 private void btnCleanHistory_Click(object sender, RoutedEventArgs e)

660 {

661 MessageBoxResult resp = MessageBox.Show("Deseja limpar as pesquisas efetuadas
anteriormente?

", "Limpar Histórico", MessageBoxButton.YesNo, MessageBoxImage.Question,
MessageBoxResult.No);

662 if (resp == MessageBoxResult.Yes)
```

```
663 {
664 //Ligação à base de dados através de um TableAdapter
665 myQueriesTableAdapter mqta = new myQueriesTableAdapter();
666 //Alteração da ConnectionString com base no que está configurado no ficheiro .config
667
mqta.ChangeConnection(ConfigurationManager.ConnectionStrings["BooksDatabase.Properti
es.
Settings.BooksLibraryConnectionString"].ConnectionString);
668
669 //Eliminação de todas as pesquisas efetuadas pelo utilizador
670 int retx = mqta.DeleteAllUserSearches(Utils.CurrentUserID);
671 if (retx >= 1)
672 {
673 //Se eliminou com sucesso, limpar o controlo das pesquisas efetuadas
674 lstLastSearches.Items.Clear();
675 }
676 }
677 }
678 private void btnDownloadPDF_Click(object sender, RoutedEventArgs e)
```

```
679 {
680 //Se não existe nenhum livro selecionado sair da função
681 if (cmbBooks.SelectedItem == null)
682 return;
683
684 //Obter o ID do livro selecionado
685 int book_id = ((KeyValuePair<string, int>)cmbBooks.SelectedItem).Value;
686
687 //Ligação à base de dados através de um TableAdapter
688 BooksTableAdapters.TBooksTableAdapter tbta = new
BooksTableAdapters.TBooksTableAdapter();
689 tbta.Connection.ConnectionString =
ConfigurationManager.ConnectionStrings["BooksDatabase.
Properties.Settings.BooksLibraryConnectionString"].ConnectionString;
690 //Carregamento da tabela
691 Books.TBooksDataTable tbdt = tbta.GetData_TBooks(book_id, null, null, null, null,
null);
692
693 //Se encontrou um registo
```

```
694 if (tbdt != null && tbdt.Rows.Count > 0)
695 {
696 Books.TBooksRow row = ((Books.TBooksRow)tbdt.Rows[0]);
697
698 //Se tem um ficheiro em PDF
699 if (row.IsbookNull() == false)
700 {
701 //Pedir para gravar o PDF onde quiser
702 SaveFileDialog sfd = new SaveFileDialog();
703 sfd.CheckPathExists = true;
704 sfd.Filter = Utils.AdobeAcrobatFilter;
705 sfd.Title = "Onde deseja guardar o livro?";
706 sfd.OverwritePrompt = true;
707
708 //Atribuir uma sugestão de nome para a gravação do ficheiro
709 string filename = row.title + " (" + row.author + " - " + Utils.
GetPublisherDescription(row.publisher) + ")";
710 filename = Utils.SetGoodFilename(filename);
```

```
711 sfd.FileName = filename;

712 if (sfd.ShowDialog() == true)

713 {

714 if (Utils.ByteArrayToFile(sfd.FileName, row.book))

715 {

716 //Ficheiro guardado

717 }

718 }

719 }

720 }

721 }

722 private void btnUploadPDF_Click(object sender, RoutedEventArgs e)

723 {

724 //Ler o ficheiro obtido do disco

725 byte[] pdfbyte = Utils.GetFileBytes(null, Utils.AdobeAcrobatFilter, "Escolher um PDF
do

livro");

726
```

```
727 //Se o ficheiro foi lido

728 if (pdfbyte != null)

729 {

730 //Obter o ID do livro selecionado

731 int book_id = ((KeyValuePair<string, int>)cmbBooks.SelectedItem).Value;

732

733 //Ligação à base de dados através de um TableAdapter

734 BooksTableAdapters.TBooksTableAdapter tbta = new
BooksTableAdapters.TBooksTableAdapter()

;

735 tbta.Connection.ConnectionString =
ConfigurationManager.ConnectionStrings["BooksDatabase

.Properties.Settings.BooksLibraryConnectionString"].ConnectionString;

736 //Carregamento da tabela

737 Books.TBooksDataTable tbdt = tbta.GetData_TBooks(book_id, null, null, null, null,
null);

738

739 //Se encontrou o registo

740 if (tbdt != null && tbdt.Rows.Count > 0)
```

```
741 {
742 Books.TBooksRow row = ((Books.TBooksRow)tbdt.Rows[0]);
743 if (row != null)
744 {
745 //Atribuir o ficheiro PDF ao campo
746 row.book = pdfbyte;
747
748 //Atualizar a base de dados
749 int ret = tbta.Update(row);
750 if (ret == 1)
751 {
752 //Existe um PDF
753 hasBookPDF = true;
754 //Disponibilizar o botão de Download do PDF
755 btnDownloadPDF.IsEnabled = true;
756 }
757 }
758 }
```

```
759 }
760
761 //Libertar memória
762 pdfbyte = null;
763 }
764 private void btnManagePublishers_Click(object sender, RoutedEventArgs e)
765 {
766 //Abrir a janela de gestão de Editoras
767 ManagePublishers manPubs = new ManagePublishers();
768 manPubs.Owner = this;
769 manPubs.ShowDialog();
770
771 //Recarregar as editoras
772 LoadPublishers();
773 }
774 #endregion
775
776 #region Combos
```

```
777 private void cmbBooks_SelectionChanged(object sender, SelectionChangedEventArgs e)
778 {
779 //Obter o índice do item da combo dos livros
780 current_book_id = cmbBooks.SelectedIndex;
781
782 //Carregar o livro selecionado
783 LoadCurrentSelectedBook();
784
785 //Habilitar/desabilitar controlos baseado nas variáveis 'mode' e 'Utils.UserType'
786 SetEnvironment();
787 }
788 #endregion
789
790 #region Functions & Methods
791 private void FillComboOfBooksFound()
792 {
793 //Guardar o modo para mais tarde o voltar a obter
794 enmAdminMode old_mode = mode;
```

795

796 //Ligação à base de dados através de um TableAdapter

797 BooksTableAdapters.Books_FindByAnythingTableAdapter bfbata = new BooksTableAdapters.

Books_FindByAnythingTableAdapter();

798 bfbata.Connection.ConnectionString = ConfigurationManager.ConnectionStrings["BooksDatabase.

Properties.Settings.BooksLibraryConnectionString"].ConnectionString;

799 //Carregamento da tabela

800 Books.Books_FindByAnythingDataTable bfbadt = bfbata.GetData_BooksBySearch("%" + txtSearch.

Text + "%");

801

802 //Carregamento do controlo

803 cmbBooks.Items.Clear();

804 cmbBooks.DisplayMemberPath = "Key";

805 cmbBooks.SelectedValuePath = "Value";

806 foreach (Books.Books_FindByAnythingRow row in bfbadt.Rows)

807 cmbBooks.Items.Add(new KeyValuePair<string, int>(row.title, row.id));

```
808

809 //Contagem do total de livros encontrados

810 total_rows_found = bfbadt.Rows.Count;

811 //Índice atual caso tenha ou não encontrado registos

812 current_book_id = (bfbadt.Rows.Count >= 1 ? 0 : -1);

813 loadingCombo = true;

814

815 //Selecionar o 1º livro encontrado

816 cmbBooks.SelectedIndex = current_book_id;

817

818 //Recolocar o modo guardado

819 mode = old_mode;

820

821 //Habilitar/desabilitar controlos baseado nas variáveis 'mode' e 'Utils.UserType'

822 SetEnvironment();

823 }

824 /// <summary>

825 /// Habilitar/desabilitar controlos baseado nas variáveis 'mode' e 'Utils.UserType'
```

```
826 /// </summary>
827 private void SetEnvironment()
828 {
829 if (loading_window) return;
830
831 dpSearch.IsEnabled = (mode == enmAdminMode.Search || mode == enmAdminMode.Navigate);
832 btnSearch.IsEnabled = (mode == enmAdminMode.Search || mode == enmAdminMode.Navigate);
833
834 dpBooksFound.IsEnabled = (total_rows_found > 0) && (mode == enmAdminMode.Search || mode ==
enmAdminMode.Navigate);
835 dpCancel.Visibility = ((mode == enmAdminMode.EditBook || mode == enmAdminMode.NewBook) ?
System.Windows.Visibility.Visible : System.Windows.Visibility.Hidden);
836
837 //Se o utilizador é administrador
838 if (Utils.UserType == 1)
839 {
```

```
840 if (total_rows_found == 0)

841 {

842 //Se não foram encontrados registos

843 dpReference.IsEnabled = (mode == enmAdminMode.NewBook || mode ==
enmAdminMode.

EditBook);

844 dpTitle.IsEnabled = (mode == enmAdminMode.NewBook || mode ==
enmAdminMode.EditBook);

845 dpAuthor.IsEnabled = (mode == enmAdminMode.NewBook || mode ==
enmAdminMode.EditBook)

;

846 dpPublisher.IsEnabled = (mode == enmAdminMode.NewBook || mode ==
enmAdminMode.

EditBook);

847 lblResumeOfBook.IsEnabled = (mode == enmAdminMode.NewBook || mode ==
enmAdminMode.

EditBook);

848 txtBookDescription.IsEnabled = (mode == enmAdminMode.NewBook || mode ==
enmAdminMode

.EditBook);

849 }
```

```
850 else
851 {
852 //Se foram encontrados registos
853 dpReference.IsEnabled = true;
854 dpTitle.IsEnabled = true;
855 dpAuthor.IsEnabled = true;
856 dpPublisher.IsEnabled = true;
857 lblResumeOfBook.IsEnabled = true;
858 txtBookDescription.IsEnabled = true;
859 }
860 }
861 else //Se o utilizador não é administrador
862 {
863 dpReference.IsEnabled = false;
864 dpTitle.IsEnabled = false;
865 dpAuthor.IsEnabled = false;
866 dpPublisher.IsEnabled = false;
867 lblResumeOfBook.IsEnabled = false;
```

```
868 txtBookDescription.IsEnabled = false;

869 }

870

871 btnManagePublishers.IsEnabled = dpPublisher.IsEnabled;

872

873 btnNewBook.IsEnabled = (mode == enmAdminMode.Search || mode ==
enmAdminMode.Navigate);

874 btnSaveBook.IsEnabled = (mode == enmAdminMode.NewBook || mode ==
enmAdminMode.EditBook);

875

876 btnPrevious.IsEnabled = (current_book_id > 0) && (mode == enmAdminMode.Search ||
mode ==

enmAdminMode.Navigate);

877 btnNext.IsEnabled = (current_book_id < total_rows_found - 1) && (mode ==
enmAdminMode.Search

|| mode == enmAdminMode.Navigate);

878

879 btnUpload.IsEnabled = (total_rows_found > 0) && (mode == enmAdminMode.Search ||
mode ==

enmAdminMode.Navigate);
```

880

881 //Controlos indisponíveis para Utilizadores normais...

882 btnNewBook.Visibility = ((Utils.UserType == 1) ? System.Windows.Visibility.Visible : System.

Windows.Visibility.Hidden);

883 btnSaveBook.Visibility = ((Utils.UserType == 1) ? System.Windows.Visibility.Visible : System

.Windows.Visibility.Hidden);

884 btnUpload.Visibility = ((Utils.UserType == 1) ? System.Windows.Visibility.Visible : System.

Windows.Visibility.Hidden);

885 btnManageUsers.Visibility = ((Utils.UserType == 1) ? System.Windows.Visibility.Visible :

System.Windows.Visibility.Hidden);

886

887 btnDownloadPDF.IsEnabled = (total_rows_found > 0) && (mode == enmAdminMode.Search || mode ==

enmAdminMode.Navigate) && hasBookPDF;

888 btnUploadPDF.IsEnabled = (total_rows_found > 0) && (mode == enmAdminMode.Search || mode ==

enmAdminMode.Navigate);

```
889 btnUploadPDF.Visibility = ((Utils.UserType == 1) ? System.Windows.Visibility.Visible
:
System.Windows.Visibility.Hidden);
890
891 //Se não foi possível encontrar livros
892 if (total_rows_found == 0)
893 {
894 //Mostrar que não foram encontrados registos
895 lblRecords.Content = "Registo nº 0 de 0";
896 }
897 else
898 {
899 //Se estamos no modo de criar novo livro, mostrar que não foram encontrados registos
900 if (mode == enmAdminMode.NewBook)
901 lblRecords.Content = "Registo nº 0 de 0";
902 else //Mostrar quantos registos existem e qual estamos a ver de momento
903 lblRecords.Content = string.Format("Registo nº {0} de {1}", current_book_id + 1,
total_rows_found);
```

```
904 }

905 }

906 /// <summary>

907 /// Verificar se o controlo RichTextBox está vazio

908 /// </summary>

909 /// <param name="rtb">Controlo do Tipo RichTextBox</param>

910 /// <returns>true se está vazio; false se tem texto</returns>

911 private bool IsRTBEmpty(RichTextBox rtb)

912 {

913 string text = new TextRange(rtb.Document.ContentStart,
rtb.Document.ContentEnd).Text;

914 return !String.IsNullOrWhiteSpace(text);

915 }

916 #endregion

917

918 #region TextChanged

919 private void txtReference_TextChanged(object sender, TextChangedEventArgs e)

920 {
```

```
921 //Se o modo é diferente de novo livro
922 if (mode != enmAdminMode.NewBook)
923 mode = enmAdminMode.EditBook; //Colocar o modo como edição
924
925 //Habilitar/desabilitar controlos baseado nas variáveis 'mode' e 'Utils.UserType'
926 SetEnvironment();
927 }
928 private void txtTitle_TextChanged(object sender, TextChangedEventArgs e)
929 {
930 //Se o modo é diferente de novo livro
931 if (mode != enmAdminMode.NewBook)
932 mode = enmAdminMode.EditBook; //Colocar o modo como edição
933
934 //Habilitar/desabilitar controlos baseado nas variáveis 'mode' e 'Utils.UserType'
935 SetEnvironment();
936 }
937 private void txtAuthor_TextChanged(object sender, TextChangedEventArgs e)
938 {
```

```
939 //Se o modo é diferente de novo livro
940 if (mode != enmAdminMode.NewBook)
941 mode = enmAdminMode.EditBook; //Colocar o modo como edição
942
943 //Habilitar/desabilitar controlos baseado nas variáveis 'mode' e 'Utils.UserType'
944 SetEnvironment();
945 }
946 private void txtBookDescription_TextChanged(object sender, TextChangedEventArgs e)
947 {
948 //Se o modo é diferente de novo livro
949 if (mode != enmAdminMode.NewBook)
950 mode = enmAdminMode.EditBook; //Colocar o modo como edição
951
952 //Habilitar/desabilitar controlos baseado nas variáveis 'mode' e 'Utils.UserType'
953 SetEnvironment();
954 }
955 private void cmbPublisher_SelectionChanged(object sender,
SelectionChangedEventArgs e)
```

```
956 {
957 //Se o modo é diferente de novo livro
958 if (mode != enmAdminMode.NewBook)
959 mode = enmAdminMode.EditBook; //Colocar o modo como edição
960
961 //Habilitar/desabilitar controlos baseado nas variáveis 'mode' e 'Utils.UserType'
962 SetEnvironment();
963 }
964 #endregion
965 }
966 }
```

Anexo III – APP.config

```
1 <?xml version="1.0" encoding="utf-8" ?>

2 <configuration>

3 <configSections>

4 </configSections>

5 <connectionStrings>

6 <add name="BooksDatabase.Properties.Settings.BooksLibraryConnectionString"

7 connectionString="Password=sapwd;Persist Security Info=True;User ID=sa;Initial
Catalog=

BooksLibrary_AUX;Data Source=(local)"

8 providerName="System.Data.SqlClient" />

9 </connectionStrings>

10 </configuration>
```

Anexo IV – APP.xaml

```
1 <Application x:Class="BookLibrary.App"

2 xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"

3 xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"

4 StartupUri="Login.xaml">

5 <!-- AdminView -->

6 <Application.Resources>

7

8 </Application.Resources>

9 </Application>

10
```

Anexo V – login.xaml

```
1 using System;

2 using System.Collections.Generic;

3 using System.Configuration;

4 using System.Linq;

5 using System.Text;

6 using System.Threading.Tasks;

7 using System.Windows;

8 using System.Windows.Controls;

9 using System.Windows.Data;

10 using System.Windows.Documents;

11 using System.Windows.Input;

12 using System.Windows.Media;

13 using System.Windows.Media.Imaging;

14 using System.Windows.Navigation;

15 using System.Windows.Shapes;

16
```

```
17 namespace istec.pg
18 {
19 /// <summary>
20 /// Interaction logic for MainWindow.xaml
21 /// </summary>
22 public partial class Login : Window
23 {
24 /// <summary>
25 /// Construtor da janela
26 /// </summary>
27 public Login()
28 {
29 InitializeComponent();
30
31 //Colocar o foco na caixa de texto do utilizador
32 txtUsername.Focus();
33 }
34
```

```
35 private void btnLogin_Click(object sender, RoutedEventArgs e)
36 {
37 try
38 {
39 //Ligação à base de dados através de um TableAdapter
40 UsersTableAdapters.TUsersTableAdapter tuta = new
UsersTableAdapters.TUsersTableAdapter();
41 tuta.Connection.ConnectionString =
ConfigurationManager.ConnectionStrings["BooksDatabase.
Properties.Settings.BooksLibraryConnectionString"].ConnectionString;
42 //Carregamento da tabela
43 Users.TUsersDataTable tudt = tuta.GetData_TUsers(null, txtUsername.Text, txtPassword.
Password, null, null);
44
45 //Se existe o utilizador e a password é a correta
46 if (tudt.Rows.Count == 1)
47 {
48 Users.TUsersRow userRow = ((Users.TUsersRow)tudt.Rows[0]);
49
```

```
50 //Ligação à base de dados através de um TableAdapter
51 UsersTableAdapters.TUsersTableAdapter tuta2 = new UsersTableAdapters.
TUsersTableAdapter();
52 tuta2.Connection.ConnectionString = ConfigurationManager.ConnectionStrings[
"BooksDatabase.Properties.Settings.BooksLibraryConnectionString"].ConnectionString;
53 //Carregamento da tabela
54 Users.TUsersDataTable tudt2 = tuta2.GetData_TUsers(userRow.id, null, null, null,
null);
55
56 Users.TUsersRow userUpdateDate = ((Users.TUsersRow)tudt2.Rows[0]);
57 //Atribuir a data do último login ao utilizador
58 userUpdateDate.last_login = DateTime.Now;
59
60 //Atualizar a base de dados
61 int ret = tuta2.Update(userUpdateDate);
62 if (ret > 0)
63 {
64 //Atribuir as informações do utilizador às variáveis globais
```

```
65 Utils.CurrentUserID = userRow.id;

66 Utils.CurrentUsername = userRow.username;

67 Utils.LastLogin = userUpdateDate.last_login;

68 Utils.UserType = userRow.user_type;

69

70 //Mostrar a janela principal

71 AdminView newAdminView = new AdminView();

72 newAdminView.Show();

73 this.Close();

74 }

75 }

76 else

77 {

78 //Erro a iniciar sessão / Utilizador ou password errados

79 MessageBoxResult result = MessageBox.Show("Erro a iniciar sessão", "Erro",

MessageBoxButton.OK, MessageBoxImage.Exclamation);

80 txtPassword.Password = string.Empty;

81 txtPassword.Focus();
```

```
82 }

83 }

84 catch (Exception ex)

85 {

86 MessageBox.Show("Erro no início de sessão.\r\nErro: " + ex.Message, "Erro a iniciar

sessão", MessageBoxButton.OK, MessageBoxImage.Exclamation);

87 }

88 }

89 private void btnExit_Click(object sender, RoutedEventArgs e)

90 {

91 //Sair da aplicação

92 this.Close();

93 }

94 }

95 }
```

Anexo VI - ManagePublishers.xaml

```
1 using Microsoft.Win32;

2 using System;

3 using System.Collections.Generic;

4 using System.Configuration;

5 using System.IO;

6 using System.Linq;

7 using System.Text;

8 using System.Windows;

9 using System.Windows.Controls;

10 using System.Windows.Data;

11 using System.Windows.Documents;

12 using System.Windows.Input;

13 using System.Windows.Media;

14 using System.Windows.Media.Imaging;

15 using System.Windows.Shapes;

16
```

```
17 namespace istec.pg
18 {
19 /// <summary>
20 /// Interaction logic for AdminView.xaml
21 /// </summary>
22 public partial class ManagePublishers : Window
23 {
24 #region Window Methods
25 /// <summary>
26 /// Construtor da janela
27 /// </summary>
28 public ManagePublishers()
29 {
30 InitializeComponent();
31 }
32 private void Window_Loaded(object sender, RoutedEventArgs e)
33 {
34 //Carregar as editoras
```

```
35 LoadPublishers();
36 }
37 #endregion
38
39 #region Loadings...
40 /// <summary>
41 /// Carregar as editoras
42 /// </summary>
43 private void LoadPublishers()
44 {
45 //Ligação à base de dados através de um TableAdapter
46 PublishersTableAdapters.TPublishersTableAdapter tuta = new PublishersTableAdapters.
TPublishersTableAdapter();
47                    tuta.Connection.ConnectionString                    =
ConfigurationManager.ConnectionStrings["BooksDatabase.
Properties.Settings.BooksLibraryConnectionString"].ConnectionString;
48 //Carregamento da tabela
49 Publishers.TPublishersDataTable tudt = tuta.GetData_TPublishers(null, null);
```

```
50

51 //Limpar a datagrid

52 dgPublishers.Items.Clear();

53

54 //Percorrer as editoras para as colocar na datagrid

55 foreach (Publishers.TPublishersRow row in tudt.Rows)

56 {

57 //carregar uma class de apoio à criação do registo para a datagrid

58 var new_Publisher = new Publisher

59 {

60 ID = Convert.ToInt32(row.ItemArray[0]),

61 Description = row.ItemArray[1].ToString()

62 };

63

64 //adicionar o registo à datagrid

65 dgPublishers.Items.Add(new_Publisher);

66 }

67 }
```

```
68 #endregion

69

70 #region DataGrid Functions

71 private void dgPublishers_SelectionChanged(object sender, SelectionChangedEventArgs
e)

72 {

73 //Limpar os controlos

74 emptyControls();

75

76 //Se uma das linhas da grelhas estiver selecionada

77 if (dgPublishers.SelectedItem != null)

78 {

79 //Obter a linha selecionada

80 var Publisher_selected = ((Publisher)dgPublishers.SelectedItem);

81

82 //Se existir

83 if (Publisher_selected != null)

84 {
```

```
85 //Preencher os controlos com os valores
86 txtID.Text = Publisher_selected.ID.ToString();
87 txtPublisher.Text = Publisher_selected.Description;
88 }
89 }
90 }
91 private void dgPublishers_MouseLeftButtonUp(object sender, MouseButtonEventArgs e)
92 {
93 //Validar se o utilizador clicou numa linha com o rato
94 var hit = VisualTreeHelper.HitTest((Visual)sender,
e.GetPosition((IInputElement)sender));
95 DependencyObject cell = VisualTreeHelper.GetParent(hit.VisualHit);
96 while (cell != null && !(cell is System.Windows.Controls.DataGridCell)) cell =
VisualTreeHelper.GetParent(cell);
97 System.Windows.Controls.DataGridCell targetCell = cell as System.Windows.Controls.
DataGridCell;
98
99 //Código para quando se seleciona ou não uma linha da datagrid
```

```
100 SelectedOrNotRowCode(targetCell);

101 }

102 private void dgPublishers_MouseRightButtonUp(object sender, MouseButtonEventArgs
e)

103 {

104 //Validar se o utilizador clicou numa linha com o rato

105 var hit = VisualTreeHelper.HitTest((Visual)sender,
e.GetPosition((IInputElement)sender));

106 DependencyObject cell = VisualTreeHelper.GetParent(hit.VisualHit);

107 while (cell != null && !(cell is System.Windows.Controls.DataGridCell)) cell =

VisualTreeHelper.GetParent(cell);

108 System.Windows.Controls.DataGridCell targetCell = cell as System.Windows.Controls.

DataGridCell;

109

110 //Código para quando se seleciona ou não uma linha da datagrid

111 SelectedOrNotRowCode(targetCell);

112 }

113 #endregion

114
```

```
115 #region Functions

116 /// <summary>

117 /// Limpar os controlos

118 /// </summary>

119 private void emptyControls()

120 {

121 //Limpar os controlos

122 txtID.Text = string.Empty;

123 txtPublisher.Text = string.Empty;

124 }

125 private void SelectedOrNotRowCode(System.Windows.Controls.DataGridCell
targetCell)

126 {

127 //Se não existe nenhuma linha selecionada

128 if (targetCell == null)

129 {

130 //Limpar os controlos

131 emptyControls();
```

```
132

133 //Desselecionar a linha da datagrid

134 dgPublishers.SelectedItems.Clear();

135 }

136

137 //Disponibilizar ou não os controlos de alterar e remover caso esteja ou não selecionada

138 btnChange.IsEnabled = (targetCell != null);

139 btnRemove.IsEnabled = (targetCell != null);

140 }

141 #endregion

142

143 #region Buttons

144 private void btnAdd_Click(object sender, RoutedEventArgs e)

145 {

146 //Ligação à base de dados através de um TableAdapter

147 PublishersTableAdapters.TPublishersTableAdapter tuta = new PublishersTableAdapters.

TPublishersTableAdapter();
```

```
148                 tuta.Connection.ConnectionString =
ConfigurationManager.ConnectionStrings["BooksDatabase.

Properties.Settings.BooksLibraryConnectionString"].ConnectionString;

149 //Carregamento da tabela

150     Publishers.TPublishersDataTable tudt = tuta.GetData_TPublishers(null,
txtPublisher.Text);

151

152 //Se a editora já existe

153 if (tudt.Rows.Count > 0)

154 {

155 MessageBox.Show("A editora a adicionar já existe na base de dados.", "Editora já
existe"

, MessageBoxButton.OK, MessageBoxImage.Exclamation);

156 return;

157 }

158

159 //Criar um novo registo

160 Publishers.TPublishersRow row = tudt.NewTPublishersRow();

161
```

```
162 //Atribuir o nome da editora ao campo
163 row.description = txtPublisher.Text;
164
165 //Adicionar o registo à tabela
166 tudt.Rows.Add(row);
167
168 //Atualizar a base de dados
169 int ret = tuta.Update(tudt);
170 if (ret > 0)
171 {
172 //Carregar as editoras
173 LoadPublishers();
174 }
175 }
176 private void btnChange_Click(object sender, RoutedEventArgs e)
177 {
178 //Ligação à base de dados através de um TableAdapter
```

```
179 PublishersTableAdapters.TPublishersTableAdapter tuta_check = new
PublishersTableAdapters.
TPublishersTableAdapter();
180 tuta_check.Connection.ConnectionString = ConfigurationManager.ConnectionStrings[
"BooksDatabase.Properties.Settings.BooksLibraryConnectionString"].ConnectionString;
181 //Carregamento da tabela
182 Publishers.TPublishersDataTable tudt_check = tuta_check.GetData_TPublishers(null,
txtPublisher.Text);
183
184 //Se a editora já existe
185 if (tudt_check.Rows.Count > 0)
186 {
187 MessageBox.Show("A editora a alterar já existe na base de dados.", "Editora já existe",
MessageBoxButton.OK, MessageBoxImage.Exclamation);
188 return;
189 }
190
191 //Ligação à base de dados através de um TableAdapter
```

```
192 PublishersTableAdapters.TPublishersTableAdapter tuta = new PublishersTableAdapters.
TPublishersTableAdapter();
193tuta.Connection.ConnectionString=ConfigurationManager.ConnectionStrings["BooksDat
abase.Properties.Settings.BooksLibraryConnectionString"].ConnectionString;
194 //Carregamento da tabela
195 Publishers.TPublishersDataTable tudt =
tuta.GetData_TPublishers(Convert.ToInt32(txtID.Text),
null);
196
197 //Se a editora não existe
198 if (tudt.Rows.Count == 0)
199 {
200 MessageBox.Show("A editora a alterar não existe na base de dados.", "Editora não
existe"
, MessageBoxButton.OK, MessageBoxImage.Exclamation);
201 return;
202 }
203
204 //Obter o registo
```

```
205 Publishers.TPublishersRow row = ((Publishers.TPublishersRow)tudt.Rows[0]);
206
207 if (row != null)
208 {
209 //Atribuir o nome da editora ao campo
210 row.description = txtPublisher.Text;
211
212 //Atualizar a base de dados
213 int ret = tuta.Update(row);
214 if (ret > 0)
215 {
216 //Carregar as editoras
217 LoadPublishers();
218 }
219 }
220 }
221 private void btnRemove_Click(object sender, RoutedEventArgs e)
222 {
```

```
223 //Ligação à base de dados através de um TableAdapter

224 BooksTableAdapters.TBooksTableAdapter tbta = new
BooksTableAdapters.TBooksTableAdapter();

225 tbta.Connection.ConnectionString =
ConfigurationManager.ConnectionStrings["BooksDatabase.

Properties.Settings.BooksLibraryConnectionString"].ConnectionString;

226 //Carregamento da tabela

227 Books.TBooksDataTable tbdt = tbta.GetData_TBooks(null, null, null, null,
Convert.ToInt32

(txtID.Text), null);

228

229 //Se a editora está a ser utilizada num livro

230 if (tbdt.Rows.Count > 0)

231 {

232 //Não deixar eliminar porque iria dar erro

233 MessageBox.Show("A editora a remover está a ser utilizada e não pode ser removida.",

"Editora não pode ser removida", MessageBoxButton.OK, MessageBoxImage.Exclamation);

234 return;

235 }
```

```
236

237 //Ligação à base de dados através de um TableAdapter

238 PublishersTableAdapters.TPublishersTableAdapter tuta = new PublishersTableAdapters.

TPublishersTableAdapter();

239                    tuta.Connection.ConnectionString                    =
ConfigurationManager.ConnectionStrings["BooksDatabase.

Properties.Settings.BooksLibraryConnectionString"].ConnectionString;

240 //Carregamento da tabela

241          Publishers.TPublishersDataTable          tudt          =
tuta.GetData_TPublishers(Convert.ToInt32(txtID.Text),

null);

242

243 //Se a editora não existe

244 if (tudt.Rows.Count == 0)

245 {

246 MessageBox.Show("A editora a remover não existe na base de dados.", "Editora não
existe"

, MessageBoxButton.OK, MessageBoxImage.Exclamation);

247 return;
```

```
248 }

249

250 //Obter o registo

251 Publishers.TPublishersRow row = ((Publishers.TPublishersRow)tudt.Rows[0]);

252

253 if (row != null)

254 {

255 MessageBoxResult resp = MessageBox.Show("Deseja remover a editora selecionada?",

"Remover Editora", MessageBoxButton.YesNo, MessageBoxImage.Question,
MessageBoxResult.No);

256

257 if (resp == MessageBoxResult.Yes)

258 {

259 //Eliminar o registo

260 row.Delete();

261

262 //Atualizar a base de dados

263 int ret = tuta.Update(row);
```

```
264 if (ret > 0)
265 {
266 //Carregar as editoras
267 LoadPublishers();
268 }
269 }
270 }
271 }
272 private void btnClose_Click(object sender, RoutedEventArgs e)
273 {
274 //Sair desta janela
275 this.Close();
276 }
277 #endregion
278 }
279
280 /// <summary>
281 /// Class de Editoras
```

```
282 /// </summary>
283 public class Publisher
284 {
285 #region Variáveis
286 private int _ID;
287 private string _Description;
288 #endregion
289
290 #region Propriedades
291 public int ID
292 {
293 get { return _ID; }
294 set { _ID = value; }
295 }
296 public string Description
297 {
298 get { return _Description; }
299 set { _Description = value; }
```

```
300 }
301 #endregion
302
303 /// <summary>
304 /// Construtor da Class
305 /// </summary>
306 public Publisher()
307 {
309 }
310
311 /// <summary>
312 /// Construtor da Class com inicialização de variáveis
313 /// </summary>
314 /// <param name="id">ID da editora</param>
315 /// <param name="description">Descrição no nome da editora</param>
316 public Publisher(int id, string description)
317 {
318 this.ID = id;
```

```
319 this.Description = description;

320 }

321 }

322 }
```

Anexo VII - ManageUsers.xaml

```
1 using Microsoft.Win32;

2 using System;

3 using System.Collections.Generic;

4 using System.Configuration;

5 using System.IO;

6 using System.Linq;

7 using System.Text;

8 using System.Windows;

9 using System.Windows.Controls;

10 using System.Windows.Data;

11 using System.Windows.Documents;

12 using System.Windows.Input;

13 using System.Windows.Media;

14 using System.Windows.Media.Imaging;

15 using System.Windows.Shapes;

16
```

```
17 namespace istec.pg
18 {
19 /// <summary>
20 /// Interaction logic for AdminView.xaml
21 /// </summary>
22 public partial class ManageUsers : Window
23 {
24 #region Window Methods
25 /// <summary>
26 /// Construtor da janela
27 /// </summary>
28 public ManageUsers()
29 {
30 InitializeComponent();
31 }
32 private void Window_Loaded(object sender, RoutedEventArgs e)
33 {
34 //Carregar os tipos de utilizadores
```

```
35 LoadUserTypes();

36 //Carregar os utilizadores

37 LoadUsers();

38

39 //Imagem 'Default' da pré-visualização da photo de utilizador

40 byte[] picbyte_empty = Utils.GetFileBytes(@"~/Images/no_person_available.png",
"Image Files

(*.bmp; *.jpg; *.png; *.gif)|*.bmp; *.jpg; *.png; *.gif|Todos os ficheiros (*.*)|*.*", "Escolher
uma

imagem para o livro");

41 imgUserPhoto.Source = Utils.SetImageFromByteArray(picbyte_empty);

42 }

43 #endregion

44

45 #region Loadings...

46 /// <summary>

47 /// Carregamento dos utilizadores

48 /// </summary>

49 private void LoadUsers()
```

```
50 {
51 //Ligação à base de dados através de um TableAdapter
52 UsersTableAdapters.TUsersTableAdapter tuta = new
UsersTableAdapters.TUsersTableAdapter();
53 tuta.Connection.ConnectionString =
ConfigurationManager.ConnectionStrings["BooksDatabase.
Properties.Settings.BooksLibraryConnectionString"].ConnectionString;
54 //Carregamento da tabela
55 Users.TUsersDataTable tudt = tuta.GetData_TUsers(null, null, null, null, null);
56
57 //Limpar a datagrid dos utilizadores
58 dgUsers.Items.Clear();
59
60 //Percorrer os registos
61 foreach (Users.TUsersRow row in tudt.Rows)
62 {
63 //carregar uma class de apoio à criação do registo para a datagrid
64 var new_user = new User
65 {
```

```
66 ID = Convert.ToInt32(row.ItemArray[0]),

67 Username = row.ItemArray[1].ToString(),

68 Password = "".PadLeft(row.ItemArray[2].ToString().Length, '*'),

69 PasswordReal = row.ItemArray[2].ToString(),

70 UserType = Convert.ToInt32(row.ItemArray[3]),

71 UserTypeDesc = Utils.GetUserTypeDescription(Convert.ToInt32(row.ItemArray[3])),

72 Date = (row.ItemArray[4] == null ? "(sem data)" : row.ItemArray[4].ToString())

73 };

74

75 //adicionar o registo à datagrid

76 dgUsers.Items.Add(new_user);

77 }

78 }

79 /// <summary>

80 /// Carregamento dos tipos de utilizadores

81 /// </summary>

82 private void LoadUserTypes()

83 {
```

```
84 //Ligação à base de dados através de um TableAdapter

85 UsersTableAdapters.TUserTypesTableAdapter tpta = new UsersTableAdapters.

TUserTypesTableAdapter();

86                                    tpta.Connection.ConnectionString                    =

ConfigurationManager.ConnectionStrings["BooksDatabase.

Properties.Settings.BooksLibraryConnectionString"].ConnectionString;

87 //Carregamento da tabela

88 Users.TUserTypesDataTable tpdt = tpta.GetData_TUserTypes(null);

89

90 //Limpar a datagrid

91 cmbUsertype.Items.Clear();

92

93 //Carregamento da combo

94 cmbUsertype.DisplayMemberPath = "Key";

95 cmbUsertype.SelectedValuePath = "Value";

96 cmbUsertype.Items.Add(new KeyValuePair<string, int>(string.Empty, 0));

97 foreach (Users.TUserTypesRow row in tpdt.Rows)

98 cmbUsertype.Items.Add(new KeyValuePair<string, int>(row.description, row.id));
```

```
99
100 //Se houver pelo menos 1 registo, selecioná-lo
101 if (cmbUsertype.Items.Count > 0) cmbUsertype.SelectedIndex = 0;
102 }
103 #endregion
104
105 #region DataGrid Functions
106 private void dgUsers_SelectionChanged(object sender, SelectionChangedEventArgs e)
107 {
108 //Limpar os controlos
109 emptyControls();
110
111 //Se existe uma linha selecionada
112 if (dgUsers.SelectedItem != null)
113 {
114 //Obter o utilizador selecionado
115 var user_selected = ((User)dgUsers.SelectedItem);
116
```

```
117 if (user_selected != null)
118 {
119 //Atribuir os valores a cada controlo
120 txtID.Text = user_selected.ID.ToString();
121 txtUsername.Text = user_selected.Username;
122 txtPassword.Password = user_selected.PasswordReal;
123 Utils.SelectComboByID(cmbUsertype, user_selected.UserType);
124
125 //Obter a photo do utilizador com sessão iniciada
126 imgUserPhoto.Source = Utils.GetUserPhoto(Convert.ToInt32(txtID.Text));
127 }
128 }
129 }
130 private void dgUsers_MouseLeftButtonUp(object sender, MouseButtonEventArgs e)
131 {
132 //Validar se o utilizador clicou numa linha com o rato
133 var hit = VisualTreeHelper.HitTest((Visual)sender,
e.GetPosition((IInputElement)sender));
```

```
134 DependencyObject cell = VisualTreeHelper.GetParent(hit.VisualHit);
135 while (cell != null && !(cell is System.Windows.Controls.DataGridCell)) cell =
VisualTreeHelper.GetParent(cell);
136 System.Windows.Controls.DataGridCell targetCell = cell as System.Windows.Controls.
DataGridCell;
137
138 //Código para quando se seleciona ou não uma linha da datagrid
139 SelectedOrNotRowCode(targetCell);
140 }
141 private void dgUsers_MouseRightButtonUp(object sender, MouseButtonEventArgs e)
142 {
143 //Validar se o utilizador clicou numa linha com o rato
144 var hit = VisualTreeHelper.HitTest((Visual)sender,
e.GetPosition((IInputElement)sender));
145 DependencyObject cell = VisualTreeHelper.GetParent(hit.VisualHit);
146 while (cell != null && !(cell is System.Windows.Controls.DataGridCell)) cell =
VisualTreeHelper.GetParent(cell);
147 System.Windows.Controls.DataGridCell targetCell = cell as System.Windows.Controls.
```

```
DataGridCell;

148

149 //Código para quando se seleciona ou não uma linha da datagrid

150 SelectedOrNotRowCode(targetCell);

151 }

152 #endregion

153

154 #region Functions

155 /// <summary>

156 /// Limpar os controlos

157 /// </summary>

158 private void emptyControls()

159 {

160 txtID.Text = string.Empty;

161 txtUsername.Text = string.Empty;

162 txtPassword.Password = string.Empty;

163 Utils.SelectComboByID(cmbUsertype, 0);

164 }
```

```
165 private void SelectedOrNotRowCode(System.Windows.Controls.DataGridCell
targetCell)
166 {
167 //Se não existe nenhuma linha selecionada
168 if (targetCell == null)
169 {
170 //Limpar os controlos
171 emptyControls();
172
173 //Desselecionar a linha da datagrid
174 dgUsers.SelectedItems.Clear();
175
176 //Imagem 'Default' da pré-visualização da photo de utilizador
177 byte[] picbyte_empty = Utils.GetFileBytes(@"~/Images/no_person_available.png",
"Image
Files (*.bmp; *.jpg; *.png; *.gif)|*.bmp; *.jpg; *.png; *.gif|Todos os ficheiros (*.*)|*.*",
"Escolher uma imagem para o livro");
178 imgUserPhoto.Source = Utils.SetImageFromByteArray(picbyte_empty);
179 }
```

```
180

181 //Disponibilizar ou não os controlos de alterar e remover caso esteja ou não selecionada

182 btnChange.IsEnabled = (targetCell != null);

183 btnRemove.IsEnabled = (targetCell != null);

184 btnChangePhoto.IsEnabled = (targetCell != null);

185 }

186 #endregion

187

188 #region Buttons

189 private void btnAdd_Click(object sender, RoutedEventArgs e)

190 {

191 //Verificar os campos de introdução obrigatória

192 if (txtUsername.Text.Trim() == string.Empty ||

193 txtPassword.Password.Trim() == string.Empty ||

194 cmbUsertype.SelectedIndex == 0)

195 {

196 MessageBox.Show("Os campos são obrigatórios.", "Campos obrigatórios",
MessageBoxButton.
```

```
OK, MessageBoxImage.Exclamation);

197 return;

198 }

199

200 //Ligação à base de dados através de um TableAdapter

201 UsersTableAdapters.TUsersTableAdapter tuta = new
UsersTableAdapters.TUsersTableAdapter();

202 tuta.Connection.ConnectionString =
ConfigurationManager.ConnectionStrings["BooksDatabase.

Properties.Settings.BooksLibraryConnectionString"].ConnectionString;

203 //Carregamento da tabela

204 Users.TUsersDataTable tudt = tuta.GetData_TUsers(null, txtUsername.Text, null, null,
null);

205

206 //Se encontrou algum registo

207 if (tudt.Rows.Count > 0)

208 {

209 MessageBox.Show("O utilizador a adicionar já existe na base de dados.", "User já
existe", MessageBoxButton.OK, MessageBoxImage.Exclamation);

210 return;
```

```
211 }

212

213 //Criar um novo registo

214 Users.TUsersRow row = tudt.NewTUsersRow();

215

216 //Atribuir os valores aos campos do registo

217 row.username = txtUsername.Text;

218 row.password = txtPassword.Password;

219

220 //Obter o ID do Tipo de Utilizador

221 int usertype_id = ((KeyValuePair<string, int>)cmbUsertype.SelectedItem).Value;

222 row.user_type = usertype_id;

223 row.photo = null;

224

225 //Adicionar o registo à tabela

226 tudt.Rows.Add(row);

227

228 //Atualizar a base de dados
```

```
229 int ret = tuta.Update(tudt);

230 if (ret > 0)

231 {

232 //Carregar os utilizadores

233 LoadUsers();

234 }

235 }

236 private void btnChange_Click(object sender, RoutedEventArgs e)

237 {

238 //Ligação à base de dados através de um TableAdapter

239 UsersTableAdapters.TUsersTableAdapter tuta_check = new
UsersTableAdapters.TUsersTableAdapter

();

240 tuta_check.Connection.ConnectionString = ConfigurationManager.ConnectionStrings[

"BooksDatabase.Properties.Settings.BooksLibraryConnectionString"].ConnectionString;

241 //Carregamento da tabela

242 Users.TUsersDataTable tudt_check =
tuta_check.GetData_TUsers(Convert.ToInt32(txtID.Text),

null, null, null, null);
```

```
243

244 //Se o utilizador já existe

245 if (tudt_check.Rows.Count > 0)

246 {

247 MessageBox.Show("O utilizador a alterar já existe na base de dados.", "User não existe",

MessageBoxButton.OK, MessageBoxImage.Exclamation);

248 return;

249 }

250

251 //Ligação à base de dados através de um TableAdapter

252 UsersTableAdapters.TUsersTableAdapter tuta = new
UsersTableAdapters.TUsersTableAdapter();

253 tuta.Connection.ConnectionString =
ConfigurationManager.ConnectionStrings["BooksDatabase.

Properties.Settings.BooksLibraryConnectionString"].ConnectionString;

254 //Carregamento da tabela

255 Users.TUsersDataTable tudt = tuta.GetData_TUsers(Convert.ToInt32(txtID.Text), null,
null,

null, null);
```

```
256

257 //Se o utilizador não existe

258 if (tudt.Rows.Count == 0)

259 {

260 MessageBox.Show("O utilizador a alterar não existe na base de dados.", "User não
existe"

, MessageBoxButton.OK, MessageBoxImage.Exclamation);

261 return;

262 }

263

264 //Obter o registo

265 Users.TUsersRow row = ((Users.TUsersRow)tudt.Rows[0]);

266

267 if (row != null)

268 {

269 //Atribuir os valores aos campos do registo

270 row.username = txtUsername.Text;

271 row.password = txtPassword.Password;
```

```
272

273 //Obter o ID do Tipo de Utilizador

274 int usertype_id = ((KeyValuePair<string, int>)cmbUsertype.SelectedItem).Value;

275 row.user_type = usertype_id;

276

277 //Atualizar a base de dados

278 int ret = tuta.Update(row);

279 if (ret > 0)

280 {

281 //Carregar os utilizadores

282 LoadUsers();

283 }

284 }

285 }

286 private void btnRemove_Click(object sender, RoutedEventArgs e)

287 {

288 //Se é o último utilizador a tentar remover, não deixar

289 if (dgUsers.Items.Count <= 1)
```

```
290 {

291 MessageBox.Show("Não é possível remover todos os utilizadores.\r\nTem de existir pelo

menos um...", "Não remover todos", MessageBoxButton.OK,
MessageBoxImage.Exclamation);

292 return;

293 }

294

295 //Ligação à base de dados através de um TableAdapter

296 UsersTableAdapters.TUsersTableAdapter tuta = new
UsersTableAdapters.TUsersTableAdapter();

297 tuta.Connection.ConnectionString =
ConfigurationManager.ConnectionStrings["BooksDatabase.

Properties.Settings.BooksLibraryConnectionString"].ConnectionString;

298 //Carregamento da tabela

299 Users.TUsersDataTable tudt = tuta.GetData_TUsers(Convert.ToInt32(txtID.Text), null,
null,

null, null);

300

301 //Se o utilizador não existe

302 if (tudt.Rows.Count == 0)
```

```
303 {
304 MessageBox.Show("O utilizador a remover não existe na base de dados.", "User não existe"
, MessageBoxButton.OK, MessageBoxImage.Exclamation);
305 return;
306 }
307
308 //Obter o registo
309 Users.TUsersRow row = ((Users.TUsersRow)tudt.Rows[0]);
310
311 if (row != null)
312 {
313 MessageBoxResult resp = MessageBox.Show("Deseja remover o utilizador selecionado?",
"Remover Utilizador", MessageBoxButton.YesNo, MessageBoxImage.Question, MessageBoxResult.No);
314
315 if (resp == MessageBoxResult.Yes)
316 {
```

```
317 //Eliminar o registo
318 row.Delete();
319
320 //Atualizar a base de dados
321 int ret = tuta.Update(row);
322 if (ret > 0)
323 {
324 //Carregar os utilizadores
325 LoadUsers();
326 }
327 }
328 }
329 }
330 private void btnChangePhoto_Click(object sender, RoutedEventArgs e)
331 {
332 //Obter os bytes de uma imagem
333 byte[] picbyte = Utils.GetFileBytes(null, "Image Files (*.bmp; *.jpg; *.png; *.gif)|*.bmp;
*
```

```
.jpg; *.png; *.gif|Todos os ficheiros (*.*)|*.*", "Escolher uma imagem para o livro");

334

335 if (picbyte != null)

336 {

337 //Obter o ID do utilizador selecionado

338 int user_id = Convert.ToInt32(txtID.Text);

339

340 //Ligação à base de dados através de um TableAdapter

341 UsersTableAdapters.TUsersTableAdapter tbta = new
UsersTableAdapters.TUsersTableAdapter()

;

342 tbta.Connection.ConnectionString =
ConfigurationManager.ConnectionStrings["BooksDatabase.Properties.Settings.BooksLibrary
ConnectionString"].ConnectionString;

343 //Carregamento da tabela

344 Users.TUsersDataTable tbdt = tbta.GetData_TUsers(user_id, null, null, null, null);

345

346 if (tbdt != null && tbdt.Rows.Count > 0)

347 {
```

```
348 Users.TUsersRow row = ((Users.TUsersRow)tbdt.Rows[0]);

349

350 //Se o registo foi encontrado

351 if (row != null)

352 {

353 //Atribuir a imagem ao campo

354 row.photo = picbyte;

355

356 //Atualizar a pré-visualização do livro

357 int ret = tbta.Update(row);

358 if (ret == 1)

359 {

360 var Img = new BitmapImage();

361 Img.BeginInit();

362 Img.StreamSource = new System.IO.MemoryStream(picbyte);

363 Img.EndInit();

364

365 //Colocar a imagem no controlo
```

```
366 imgUserPhoto.Source = Img;
367 }
368 }
369 }
370 }
371
372 //Libertar memória
373 picbyte = null;
374 }
375 private void btnClose_Click(object sender, RoutedEventArgs e)
376 {
377 //Sair desta janela
378 this.Close();
379 }
380 #endregion
381 }
382
383 /// <summary>
```

```
384 /// Class de Editoras
385 /// </summary>
386 public class User
387 {
388 #region Variáveis
389 private int _ID;
390 private string _Username;
391 private string _Password;
392 private string _PasswordReal;
393 private int _UserType;
394 private string _UserTypeDesc;
395 private string _Date;
396 #endregion
397
398 #region Propriedades
399 public int ID
400 {
401 get { return _ID; }
```

```
402 set { _ID = value; }

403 }

404 public string Username

405 {

406 get { return _Username; }

407 set { _Username = value; }

408 }

409 public string Password

410 {

411 get { return _Password; }

412 set { _Password = value; }

413 }

414 public string PasswordReal

415 {

416 get { return _PasswordReal; }

417 set { _PasswordReal = value; }

418 }

419 public int UserType
```

```
420 {
421 get { return _UserType; }
422 set { _UserType = value; }
423 }
424 public string UserTypeDesc
425 {
426 get
427 {
428 return _UserTypeDesc;
429 }
430 set
431 {
432 _UserTypeDesc = value;
433 }
434 }
435 public string Date
436 {
437 get { return _Date; }
```

```
438 set { _Date = value; }
439 }
440 #endregion
441
442 /// <summary>
443 /// Construtor da Class
444 /// </summary>
445 public User()
446 {
447
448 }
449
450 /// <summary>
451 /// Construtor da Class com inicialização de variáveis
452 /// </summary>
453 /// <param name="id">ID do utilizador</param>
454 /// <param name="username">Utilizador</param>
455 /// <param name="password">Palavra-Passe do utilizador</param>
```

```
456 /// <param name="usertype">Tipo de utilizador</param>
457 /// <param name="date">Data da última sessão</param>
458 public User(int id, string username, string password, int usertype, string date)
459 {
460 this.ID = id;
461 this.Username = username;
462 this.Password = password;
463 this.UserType = usertype;
464 this.Date = date;
465 }
466 }
467 }
```

Anexo VIII - Utils.cs

```
1 using Microsoft.Win32;

2 using System;

3 using System.Collections.Generic;

4 using System.Configuration;

5 using System.IO;

6 using System.Linq;

7 using System.Text;

8 using System.Windows.Controls;

9 using System.Windows.Media;

10 using System.Windows.Media.Imaging;

11

12 namespace istec.pg

13 {

14 public static class Utils

15 {

16 /// <summary>
```

```
17 /// Variáveis privadas a esta classe
18 /// </summary>
19 #region Privates
20 private static int _CurrentUserID;
21 private static string _CurrentUsername;
22 private static DateTime _LastLogin;
23 private static int _UserType;
24 #endregion
25
26 /// <summary>
27 /// Propriedades de sessão do utilizador
28 /// </summary>
29 #region Properties
30 public static int CurrentUserID
31 {
32 get { return _CurrentUserID; }
33 set { _CurrentUserID = value; }
34 }
```

```
35 public static string CurrentUsername
36 {
37 get { return _CurrentUsername; }
38 set { _CurrentUsername = value; }
39 }
40 public static DateTime LastLogin
41 {
42 get { return _LastLogin; }
43 set { _LastLogin = value; }
44 }
45 public static int UserType
46 {
47 get { return _UserType; }
48 set { _UserType = value; }
49 }
50 #endregion
51
52 /// Variáveis para guardar os PDF's
```

```
53 public static string AdobeAcrobatFilter = "Adobe Acrobat Files (*.pdf)|*.pdf|Todos os
ficheiros

(*.*)|*.*";

54 public static string AdobeAcrobatSaveTitle = "";

55

56 /// <summary>

57 /// Função para devolver o tipo de utilizador que está com sessão iniciada

58 /// </summary>

59 /// <returns>Descrição do tipo de utilizador</returns>

60 internal static string GetCurrentUserTypeDescription()

61 {

62 //Ligação à base de dados através de um TableAdapter

63 UsersTableAdapters.TUserTypesTableAdapter tutta = new UsersTableAdapters.

TUserTypesTableAdapter();

64                                    tutta.Connection.ConnectionString                                    =
ConfigurationManager.ConnectionStrings["BooksDatabase.

Properties.Settings.BooksLibraryConnectionString"].ConnectionString;

65 //Carregamento da tabela

66 Users.TUserTypesDataTable tutdt = tutta.GetData_TUserTypes(Utils.UserType);
```

```
67

68 //Procurar o tipo de utilizador através do ID do utilizador atualmente com sessão iniciada

69 Users.TUserTypesRow rowUserType = tutdt.FindByid(Utils.UserType);

70

71 //Se encontrou devolve a descrição associada

72 if (rowUserType != null)

73 return rowUserType.description;

74 else //se não, devolve null

75 return null;

76 }

77

78 /// <summary>

79 /// Função para devolver a editora através do ID correspondente

80 /// </summary>

81 /// <param name="publisher_id">ID da editora a procurar</param>

82 /// <returns>A descrição da editora encontrada</returns>

83 internal static string GetPublisherDescription(int publisher_id)

84 {
```

```
85 //Ligação à base de dados através de um TableAdapter
86 PublishersTableAdapters.TPublishersTableAdapter tutta = new PublishersTableAdapters.
TPublishersTableAdapter();
87                    tutta.Connection.ConnectionString                    =
ConfigurationManager.ConnectionStrings["BooksDatabase.
Properties.Settings.BooksLibraryConnectionString"].ConnectionString;
88 //Carregamento da tabela
89 Publishers.TPublishersDataTable tutdt = tutta.GetData_TPublishers(publisher_id, null);
90
91 //Procurar a editora através do ID
92 Publishers.TPublishersRow rowPublisher = tutdt.FindByid(publisher_id);
93
94 //Se encontrou devolve a descrição associada
95 if (rowPublisher != null)
96 return rowPublisher.description;
97 else //se não, devolve null
98 return null;
99 }
```

```
100

101 /// <summary>

102 /// Função para devolver o tipo de utilizador com base no ID fornecido

103 /// </summary>

104 /// <param name="userType_id">ID do tipo de utilizador a obter a descrição</param>

105 /// <returns>Descrição do tipo de utilizador</returns>

106 internal static string GetUserTypeDescription(int userType_id)

107 {

108 //Ligação à base de dados através de um TableAdapter

109 UsersTableAdapters.TUserTypesTableAdapter tutta = new UsersTableAdapters.

TUserTypesTableAdapter();

110                                    tutta.Connection.ConnectionString                                    =

ConfigurationManager.ConnectionStrings["BooksDatabase.

Properties.Settings.BooksLibraryConnectionString"].ConnectionString;

111 //Carregamento da tabela

112 Users.TUserTypesDataTable tutdt = tutta.GetData_TUserTypes(userType_id);

113

114 //Procurar o tipo de utilizador através do ID
```

```
115 Users.TUserTypesRow rowUserType = tutdt.FindByid(userType_id);
116
117 //Se encontrou devolve a descrição associada
118 if (rowUserType != null)
119 return rowUserType.description;
120 else //se não, devolve null
121 return null;
122 }
123
124 /// <summary>
125 /// Função para corrigir um ficheiro com carateres inválidos num válido
126 /// </summary>
127 /// <param name="filename">Nome do ficheiro a corrigir</param>
128 /// <returns>Um ficheiro corrigido</returns>
129 internal static string SetGoodFilename(string filename)
130 {
131 string temp = filename;
132
```

```
133 //Substituição de carateres inválidos à criação de um ficheiro
134 temp = temp.Replace(@"\", string.Empty); //Substituir o carater '\' por String.Empty
135 temp = temp.Replace(@"/", string.Empty); //Substituir o carater '/' por String.Empty
136 temp = temp.Replace(@"*", string.Empty); //Substituir o carater '*' por String.Empty
137 temp = temp.Replace(@"?", string.Empty); //Substituir o carater '?' por String.Empty
138 temp = temp.Replace(@">", string.Empty); //Substituir o carater '>' por String.Empty
139 temp = temp.Replace(@"<", string.Empty); //Substituir o carater '<' por String.Empty
140 temp = temp.Replace("\"", string.Empty); //Substituir o carater '"' por String.Empty
141 temp = temp.Replace(@"|", string.Empty); //Substituir o carater '|' por String.Empty
142 temp = temp.Replace(@":", string.Empty); //Substituir o carater ':' por String.Empty
143 return temp;
144 }
145
146 /// <summary>
147 /// Método para seleccionar um item de uma combobox através do ID
148 /// </summary>
149 /// <param name="comboBox">Combobox a selecionar o item correto</param>
150 /// <param name="p">ID do item a selecionar</param>
```

```
151 internal static void SelectComboByID(ComboBox comboBox, int p)
152 {
153 //Percorrer todos os itens da combobox
154 for (int i = 0; i < comboBox.Items.Count; i++)
155 {
156 KeyValuePair<string, int> valuePair = ((KeyValuePair<string, int>)comboBox.Items[i]);
157
158 // Se o valor do item da combobox for igual ao que fornecemos
159 if (valuePair.Value == p)
160 {
161 //Selecioná-lo
162 comboBox.SelectedItem = valuePair;
163 break;
164 }
165 }
166 }
167
168 /// <summary>
```

```
169 /// Função para obter os bytes de uma imagem
170 /// Esta função usa o OpenFileDialog para o utilizador escolher qual a imagem
171 /// </summary>
172 /// <param name="imagepath">Caminho da imagem alvo</param>
173 /// <param name="filter">Filtro a usar na janela OpenFileDialog</param>
174 /// <param name="title">Título do OpenFileDialog</param>
175 /// <returns>Retorna um array de bytes da imagem</returns>
176 internal static byte[] GetFileBytes(string imagepath, string filter, string title)
177 {
178 byte[] picbyte = null;
179 bool askForImage = (imagepath == null);
180
181 if (askForImage)
182 {
183 //Abrir a janela de procura da imagem se a imagepath = null
184 OpenFileDialog ofd = new OpenFileDialog();
185 ofd.CheckFileExists = true;
186 ofd.Filter = filter;
```

```
187 ofd.Title = title;
188 if (ofd.ShowDialog() == true)
189 {
190 //Obter o caminho do ficheiro
191 imagepath = ofd.FileName;
192 }
193 }
194
195 //Se a variável imagepath não for nula nem vazia
196 if (imagepath != null && imagepath != string.Empty)
197 {
198 //Obter o caminho completo da imagem
199 imagepath = System.IO.Path.GetFullPath(imagepath);
200 //Retirar alguma configuração de caminho relativo
201 imagepath = imagepath.Replace(@"~\", string.Empty);
202
203 //Abrir a imagem e lê-la para o array de bytes
204 FileStream fs;
```

```
205 fs = new FileStream(imagepath, FileMode.Open, FileAccess.Read);
206
207 picbyte = new byte[fs.Length];
208
209 fs.Read(picbyte, 0, System.Convert.ToInt32(fs.Length));
210 fs.Close();
211 }
212 //Retornar o array de bytes
213 return picbyte;
214 }
215
216 /// <summary>
217 /// Função para obter uma imagem através de um array de bytes
218 /// Esta função usa o OpenFileDialog para o utilizador escolher qual a imagem
219 /// </summary>
220 /// <param name="picbyte">array de bytes</param>
221 /// <returns>Objeto da imagem para atribuir ao controlo Image</returns>
222 internal static ImageSource SetImageFromByteArray(byte[] picbyte)
```

```
223 {
224 var Img = new BitmapImage();
225 Img.BeginInit();
226 Img.StreamSource = new System.IO.MemoryStream(picbyte);
227 Img.EndInit();
228 return Img;
229 }
230
231 /// <summary>
232 /// Função para gravar uma imagem em disco através de um array de bytes
233 /// </summary>
234 /// <param name="_FileName">Caminho completo do ficheiro a gravar</param>
235 /// <param name="_ByteArray">Array de bytes com a imagem</param>
236 /// <returns>true - se bem gravado; false - se erro na gravação</returns>
237 internal static bool ByteArrayToFile(string _FileName, byte[] _ByteArray)
238 {
239 try
240 {
```

```
241 //Abrir ficheiro para escrita
242 System.IO.FileStream _FileStream =
243 new System.IO.FileStream(_FileName, System.IO.FileMode.Create,
244 System.IO.FileAccess.Write);
245 //Escreve um bloco de bytes usando os dados do array de bytes
246 _FileStream.Write(_ByteArray, 0, _ByteArray.Length);
247
248 //Fechar o ficheiro
249 _FileStream.Close();
250
251 return true;
252 }
253 catch (Exception _Exception)
254 {
255 //Erro
256 Console.WriteLine("Exception caught in process: {0}",
257 _Exception.ToString());
258 }
```

```
259

260 //Ocorreu um erro. Retornar false

261 return false;

262 }

263

264 /// <summary>

265 /// Obter a imagem do utilizador com sessão iniciada

266 /// </summary>

267 /// <param name="currentUserID">ID do utilizador a obter a imagem</param>

268 /// <returns>Objeto da imagem para atribuir ao controlo Image</returns>

269 internal static ImageSource GetUserPhoto(int currentUserID)

270 {

271 //Ligação à base de dados através de um TableAdapter

272 UsersTableAdapters.TUsersTableAdapter tuta = new
UsersTableAdapters.TUsersTableAdapter();

273 tuta.Connection.ConnectionString =
ConfigurationManager.ConnectionStrings["BooksDatabase.

Properties.Settings.BooksLibraryConnectionString"].ConnectionString;

274 //Carregamento da tabela através do ID do utilizador
```

```
275 Users.TUsersDataTable tudt = tuta.GetData_TUsers(currentUserID, null, null, null,
null);

276

277 //Se a pesquisa retornou algo

278 if (tudt != null && tudt.Rows.Count == 1)

279 {

280 //Obter o registo com os dados

281 Users.TUsersRow row = ((Users.TUsersRow)tudt.Rows[0]);

282

283 //Se a foto não está nula

284 if (row.IsphotoNull() == false)

285 return Utils.SetImageFromByteArray(row.photo); //Retornar a imagem através do array

de bytes

286 }

287

288 //Se não obtiver a imagem, retornar a imagem padrão

289 byte[] picbyte_empty = Utils.GetFileBytes(@"~/Images/no_person_available.png",
"Image Files
```

(*.bmp; *.jpg; *.png; *.gif)|*.bmp; *.jpg; *.png; *.gif|Todos os ficheiros (*.*)|*.*", "Escolher uma

imagem para o livro");

290 return Utils.SetImageFromByteArray(picbyte_empty); //Retornar a imagem através do array de

bytes

291 }

292

293 /// <summary>

294 ///

295 /// </summary>

296 /// <param name="id_user"></param>

297 /// <param name="id_search"></param>

298 /// <param name="dateTime"></param>

299 /// <returns></returns>

300 internal static int UserSearch_UpdateDate(int? id_user, int? id_search, DateTime dateTime)

301 {

302 if (id_search < 0)

```
303 id_search = null;

304

305 //Ligação à base de dados através de um TableAdapter

306 SearchesTableAdapters.TUsersSearchesTableAdapter tusta_aux = new
SearchesTableAdapters.

TUsersSearchesTableAdapter();

307 tusta_aux.Connection.ConnectionString = ConfigurationManager.ConnectionStrings[

"BooksDatabase.Properties.Settings.BooksLibraryConnectionString"].ConnectionString;

308 //Carregamento da tabela

309Searches.TUsersSearchesDataTabletusdt_aux=tusta_aux.GetData_TUsersSearches(id_us
er,

id_search, null);

310

311 Searches.TUsersSearchesRow userSearchRow = null;

312

313 //Se a pesquisa não tem registos

314 if (tusdt_aux != null && tusdt_aux.Rows.Count == 0)

315 userSearchRow = tusdt_aux.NewTUsersSearchesRow(); //Cria um registo vazio

316 else
```

```
317 userSearchRow = ((Searches.TUsersSearchesRow)tusdt_aux.Rows[0]); //Obtem o
registo
encontrado
318
319 if (id_search == null && Convert.ToInt32(id_search) < 0)
320 return -1;
321
322 //Atribuir os dados ao novo registo ou atualizar o registo encontrado
323 userSearchRow.id_user = Utils.CurrentUserID;
324 userSearchRow.id_search = Convert.ToInt32(id_search);
325 userSearchRow.date = DateTime.Now;
326
327 int ret2 = -1;
328 //Se a pesquisa não tem registos
329 if (tusdt_aux != null && tusdt_aux.Rows.Count == 0)
330 {
331 //Adiciona-se o novo registo
332 tusdt_aux.Rows.Add(userSearchRow);
```

```
333 //e atualiza-se na base de dados

334 ret2 = tusta_aux.Update(tusdt_aux);

335 }

336 else //Se encontrou um registo apenas se atualiza a base de dados

337 ret2 = tusta_aux.Update(userSearchRow);

338

339 //Se atualizou algo

340 if (ret2 >= 1)

341 return ret2; //Retorna a quantidade de registos (1) que atualizou

342 return -1; //Se não retorna -1

343 }

344 }

345 }
```