Instituto Superior de Tecnologias Avançadas

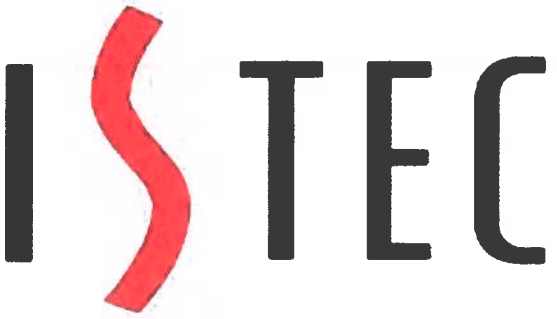
Licenciatura em Engenharia Informática I Turma 3N

Projeto Global Desenvolvido numa Biblioteca Virtual

Aluno: Ricardo Correia N:1956

Lisboa 2015/2016

13



Instituto Superior de Tecnologias Avançadas

Licenciatura em Engenharia Informática I Turma 3N

Projeto Global Desenvolvido numa Biblioteca Virtual

Trabalho realizado sob orientação de

Professor Doutor Pedro Brandão

Lisboa 2015/2016

Dedicatória

Dedico este Trabalho à Minha Querida Mulher que me Motiva Todos os Dias e a Todas as Pessoas que Querem ser Alguém Na Vida, Aprendendo.

Technology is just a tool. In Terms of getting the kids working together and motivation them, the teacher is the most important

Bill Gates

Agradecimentos

Agradeço a todos os professores que me ensinaram desde o primeiro ano, para ser alguem melhor no futuro, e não me podia esqueçer de um professor, que me fez empenhar muito para eu melhorar dizendo, aprendendo fazendo.

Agradeço do mesmo modo, a todas as pessoas que me desencorajaram por um objectivo, no qual me tinha proposto desde o primeiro dia em que entrei para o Istec fortalecendo-me mais ainda, o meu muito obrigado.

À minha mãe, que no príncipio apercebeu-se que eu iria mudar de rumo de vida, e estando ela sempre habituada a eu estar sempre junto dela, sentiu, mas o amor de uma mãe é sempre muito forte e esse estará sempre colocado no meu coração.

Imagens Ilustrativas Pág

Indice geral

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 Trabalho Digital. 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 fisica, 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 Digital Work. 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.

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.

Na 1ª Parte do projeto, "Estado De Arte", efetuou-se um estudo de todos os benefícios implícitos no mesmo como, ADO.NET, SQL Server ou Cloud Computing pretendendo demonstrar as suas melhores características e de como as mesmas em conjunto conseguiam desenvolver uma Biblioteca Virtual.

Na 2ª Parte, "A Criação das Máquinas Virtuais" é explicitada de modo a perceber passo a passo a sua configuração e implementação, com a finalidade de demonstrar todos os pré-requisitos, instalação de um controlador de domínio ou uma base de dados.

A 3ª Parte, é composta pela "Aplicação TrabDigital" que nos dá a conhecer todas as pesquisas efetuadas, autores, datas etc... .

Na 4ª Parte, iremos focar-nos no código em si que está ligado ao projeto TrabDigital.

Estado da Arte

ADO.NET

1. Introdução ao ADO.NET

As aplicações de desenvolvimento da Microsoft são conhecidas devido à sua maneira fácil de usar. Fortalecendo esta introdução o autor Vidya Agarwal diz-nos que o *ADO.NET*, permite aos programadores escreverem menos código de acesso a dados reduzindo a manutenção, e abstração (Agarwal, 2012, p. 375). O *ADO.NET* como linguagem de programação é uma escolha privilegiada no modo desconectado para o desenvolvimento web, no âmbito em que o *ADO.NET* disponibiliza um suporte para a base de dados evidenciando um cenário que demonstra vantagens, no qual se pode trabalhar em qualquer altura e ligar-se à base de dados apenas quando necessário, assim os recursos podem ser usados por outros utilizadores e aumenta a escalabilidade e desempenho das aplicações (Ferreira, 2004, p. 8). Para essas aplicações serem mais eficientes e para o rápido desenvolvimento de soluções, o *ADO.NET* utiliza o sistema (*ActiveX Data Objects*[1]) que consiste num conjunto de classes definidas pela *.net framework*[2] para acesso às bases de dados armazenadas num servidor remoto. Devido a esta tecnologia que foi desenvolvida para uma melhor integração com o *XML*[3] o *ADO.NET* tornou-se a primeira escolha para programadores permitindo-lhes trabalhar com diferentes tipos de armazenamento de dados (Hundhausen & Borg, 2002, p. 20).

[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] O .NET Framework é uma tecnologia que suporta a construção e execução de aplicações e serviços Web XML.
[3] O XML fornece acesso a uma infinidade de tecnologias para manipular, estruturar, transformar e consulta de dados.

2. A Plataforma .net

Quando a Microsoft introduziu o *.net,* criou um novo caminho para uma nova tecnologia chamada *ADO.NET*, esta linguagem contêm classes com um rico conjunto de componentes para criar aplicações distribuídas (Agarwal, 2012, p. 171) classes estas que são encontradas no *system.data.dll*[4] e estão integrados com as classes *XML*. Porém este conjunto de classes existentes na biblioteca de ficheiros especiais servem para fornecer um acesso consistente a qualquer fonte de dados, seja via *OLEDB*[5], *XML* ou *ODBC*[6] , deste modo o *.net* introduzido pela Microsoft, permitiu facilitar e poupar tempo aos programadores, tornando mais fácil e rápido estabelecerem ligações às fontes de dados de modo a recuperar, manipular e atualizar as base de dados, dados esses que podem ser internos e interagem com estruturas de dados de forma desconectada com a fonte de dados ou dados externos que estão alojados fora da aplicação numa base de dados relacional ou num ficheiro de texto.

3. Principais Vantagens do ADO.NET

O core do *ADO.NET* é o *Dataset* [7] um objeto de dados relacionais usado para manipular os dados. E este quando foi construído foi a pensar no modo desconectado (Paul Nielsen, p. 855), a razão de ser o core do *ADO.NET* significa que em qualquer altura é possível guardar um *Dataset* para a classe *XML* e assim qualquer plataforma pode devolver-nos dados de um *Dataset* do *ADO.NET* (Jason Lefebre, p. 8). Contudo além do *Dataset* no seu potente modo desconetado existe o *Datareader*[8] para o modelo conectado fazendo assim os dois tipos de componentes da arquitetura para dados do *ADO.NET* no qual existe uma biblioteca de classes que acede aos dados e que tem o nome de *.net dataproviders*[9]. Conforme o autor Jason Lefebre cita existem dois tipos de ligação a

[4] Classe do Ado.net
[5] Object Linking and Embedding Database (API Desenhada pela Microsoft)
[6] Open Database Connectivity (API para aceder a Database Management Systems)
[7] É uma ampla categoria de objetos usada para ler a partir de uma base de dados.
[8] Funciona como uma ponte entre uma base de dados e uma classe de dados desligada.
[9] É um provedor de dados usado para se conectar e executar comandos num banco de dados.

objetos o *OleDBconnection* [10] e o *Sqlconnection* [11] e são com estes dois objetos que o *ADO.NET* vai conectar para uma fonte de dados no seu modo conectado (Jason Lefebre, p. 56) ainda neste modo o *ADO.NET* suporta múltiplos *data providers* que colocam dois benefícios, um deles é que se pode programar um *data provider* específico para aceder a qualquer característica única de um SGBD[12] especial e o segundo benefício é que um específico *data provider* pode conectar diretamente para o mecanismo subjacente do SGBD em questão sem uma camada de mapeamento intermediária entre as camadas (Troelsen, p. 803).

SQL

1. Introdução ao SQL

O *SQL* [13] é uma linguagem utilizada para organizar, gerir, e devolver dados armazenados de uma base de dados, e hoje em dia é a linguagem padrão utilizada em computadores pessoais, mainframes [14], ou smartphones (James R. Groff, p. 8). Esta linguagem foi introduzida na sequência de um trabalho intitulado *"A Relational Model of Data for Large Shared Data Banks"* por Edgar Frank Codd, um investigador da IBM. A sua primeira implementação comercial do modelo relacional ficou disponível no início de 1980 desde então o modelo tem sido implementado em um largo número de sistemas comerciais (Elmasri & Navathe, p. 59) e poderá ser essa considerada uma das principais razões para o sucesso comercial, através da linguagem *SQL* pode-se alterar, expandir ou incluir de uma forma dinâmica estruturas de dados armazenados com bastante flexibilidade, mesmo quando diferentes utilizadores estão a aceder aos seus conteúdos (James R. Groff, p. 12), ou seja permite, que uma base de dados se adapte a mudanças que ocorram em aplicações *on-line* sem interrupções. Por se tratar de uma linguagem padrão, como referiu Júlio Lima, implementa os conceitos definidos no modelo

[10] É um objeto que representa uma conexão exclusiva com uma fonte de dados
[11] Representa uma sessão única de uma fonte de dados SQL Server
[12] Sistema Gestão Base de Dados
[13] Structured Query Language linguagem de pesquisa declarativa padrão para base de dados relacional
[14] Computador de grande porte, dedicado ao processamento de um enorme volume de informação

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).

2. A Importância do SQL

Com o aparecimento da *World Wide Web* [15], e a explosão da internet o *SQL* encontrou no fim de 1990 uma role standard para acesso a dados na internet (James R. Groff), e com a integração da *LAN* [16] nos escritórios por o mundo fora, um novo sistema apareceu chamado cliente/servidor (Stephens, Morgan, & Plew). Juntamente com este aparecimento a Microsoft desenvolveu ao longo do tempo desde *ODBC* [17] a *SQL/CLI* [18]o que veio a ser preponderante no mundo virtual pela simples razão que esta *API* inclui cerca de 40 API ´s diferentes, e trás vantagens que será mais fácil aceder a múltiplas base de dados com a mesma aplicação, mesmo que estejam armazenados em diferentes *DBMS*[19]. O *SQL* tem muitos benefícios, é uma linguagem fácil de aprender, tem portabilidade para outros sistemas, é uma linguagem cliente/servidor, e ainda é uma linguagem de dados de acesso á internet (N.Weinberg, p. 11). O *SQL* surgiu como uma ferramenta útil e poderosa para a ligação de pessoas, programas de computador e sistemas informáticos (James R. Groff, p. 7). Apesar de muitas tentativas para destronar o potencial do *SQL* ao longo dos anos, este manteve-se sempre extraordinariamente com sucesso para a tecnologia de informação, e no decorrer dos anos expandiu para suportar novos hardwares, novos sistemas operativos, internet, e linguagens, dominando assim na gestão de dados (James R. Groff, p. 7).

[15] Comunidade internacional de desenvolvimento
[16] Rede de área local
[17] Padrão para acesso a sistemas de dados
[18] Call Level Interface software standard para embeber SQL
[19] Tecnologia de armazenar e devolver dados

3. Vantagens Do SQL

Na linguagem SQL uma das suas maiores vantagens são os indexes[20] que sendo usados devidamente reduzem entradas e saídas de operacões no disco, aumentando bastante a performance (Pearl, p. 85), esta vai ser notória no momento em que o programador quando necessita de pesquisar dados, não é necessário efectuar uma seleção de todas as linhas das tabelas assegurando assim uma tabela mais equilibrada.

Ao contrário das linguagens procedurais[21] o SQL é uma linguagem declarativa[22] e a que tem mais sucesso a este nível, dado que o programador nas suas queries só obtem aquilo que quer.

Virtualização

1. Início da Virtualização

Em 1960 o primeiro professor de computação Christopher Strachey na Oxford University, e líder do Programming Research Group utilizou no seu paper "Time Sharing in Large Fast Computers" o termo Time Sharing, que na sua essência permitia a um programador desenvolver um programa na sua consola enquanto outro programador estava a utilizar a mesma evitando assim a habitual espera de periféricos. (Hoopes, p. 3)

O primeiro computador a trabalhar sobre o conceito multiprogramming[23] e, partilha de periféricos foi o ATLAS, baseado em um projecto desenvolvido pelo Departamento de Engenheiros Eléctricos na Universidade de Manchester. Em 1964 a *IBM* [24] lançou

[20] Pesquisa um menor numero de dados
[21] Linguagem que segue em ordem um conjunto de comandos
[22] Linguagem de alto-nível
[23] Conceito de multiplos programas serem desenvolvidos através de um único Processador
[24] International Business Machines empresa americana de informática

oficialmente as soluções de virtualização de mainframe juntamente com a nova geração de processadores sistema/360 arquitectada por Gene Amdahl[25] (Marshall & Reynolds, p. 8). Este modelo incorporava três tipos, o de multiprogramas, multiprocessos[26], e multiacessos, sendo este o modo que permitia a vários utilizadores de máquinas remotas comunicarem directamente com o sistema (Gibson, p. 61) mas o meio mais eficiente era o modo multiprogramas que providenciava mais recursos do hardware.

Este conceito foi baseado na arquitetura de uma unidade central que permitia que um sistema operativo fosse executado de uma forma transparente através de uma máquina virtual (Santana, p. 45). Para ser considerado uma máquina virtual tinha de ser simulado por um *VM/360* que continha um componente chamado *Control Program* (*CP*) denominado então de *Hypervisor* (Virtual Machine Feb. 76, p. 12) que era um sistema de controlo de programas que geravam os recursos reais de um sistema *IBM/360* no qual cada utilizador tem ao seu dispor as funções equivalentes a um *CPU*[27] real e auxiliar, armazenamento, e input/output, esta solução da *IBM* deve-se à origem de migração do sistema operacional que os seus clientes estavam a enfrentar sempre que um novo processador era lançado, a intenção era que uma unidade central poderia simultaneamente acolher diferentes versões de sistemas operacionais.

2. Os benefícios da Virtualização

Os níveis elevados de desempenho, escalabilidade ou confiabilidade são alguns dos pontos dos benefícios da virtualização, mas a prevenção para os ataques de *worms*[28], *vírus*[29], ou outros ataques de *malware* trás uma palavra-chave que é a segurança dado que acedendo virtualmente a uma máquina providencia um adicional nível de segurança quando as organizações estão a tentar prevenir esses ataques (Kusnetzry, p. 8), organizações essas que para uma melhor redução de custos através de uma localização central conseguem fazer instalações ou atualizações de software dos seus clientes

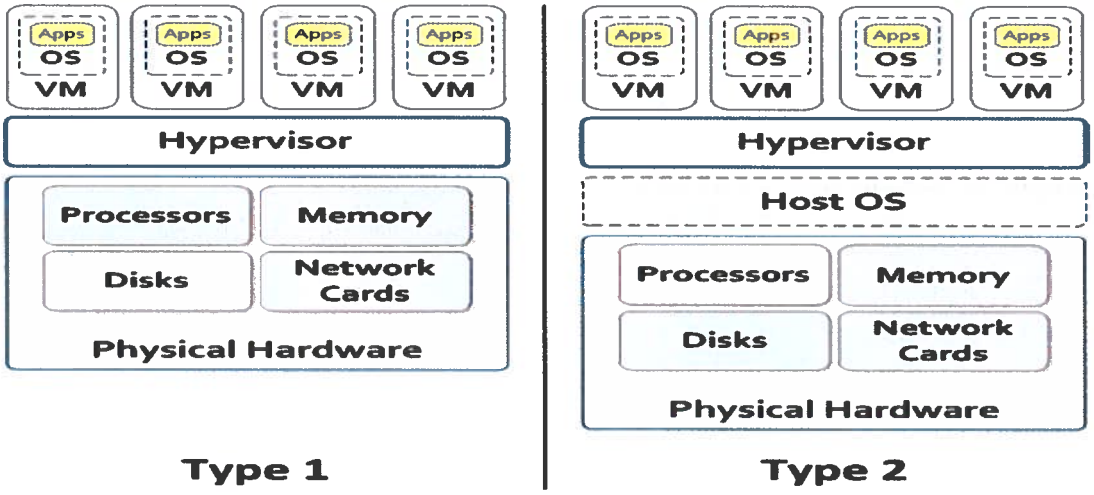
[25] Arquitecto Americano que desenvolveu o sistema/360 na IBM
[26] Executar Multiplos Processos ao mesmo tempo
[27] Unidade de processamento central
[28] Programa para tomar ações maliciosas
[29] Software para infetar o sistema

(Kusnetzry, p. 8). A virtualização tem nos dias de hoje um retorno do investimento para as grandes empresas que pode significar seis dígitos ou mais por ano, por imensas razões, seja pela redução das máquinas físicas a nível energético, ou pela redução do espaço em máquinas (Tulloch, p. 19), e de uma forma notória as empresas foram observando nos enormes proveitos que a virtualização poderia trazer entre elas o disaster recovery[30] (Goldworm, p. 5)

3. A Microsoft no mundo da Virtualização

No início de 1990 a Microsoft introduziu um novo sistema operativo de seu nome Windows NT Server, preparado para a virtualização, mas vem evoluindo desde então com o Virtual PC que era uma aplicação para os utilizadores instalarem com base no sistema operativo e correrem um segundo sistema operativo para perceberem o conceito da virtualização (Shah, p. 16). O Virtual Server foi de seguida a primeira iniciativa onde a Microsoft colocou o lado do servidor (*server - side)* que tinha uma vantagem significativa pela razão que as aplicações que possam ser incompatíveis com novas versões de um sistema operacional possam continuar a ser usadas (Shah, p. 16). O aparecimento do *hyper-v* vem com o Windows Server 2008 e foi o seu primeiro hardware de 64bits, com o *hypervisor* tipo 1 que corre diretamente no hardware que providencia uma melhor performance, disponibilidade e segurança (Tulloch, p. 23), além do desempenho no ambiente de virtualização, sendo que as máquinas virtuais não estão limitadas às limitações de um sistema operativo, ao contrário do *hypervisor* tipo 2 que corre numa camada acima de um sistema operativo reduzindo assim a performance e o numero de máquinas virtuais, nesta arquitetura a máquina virtual comunica de forma indireta com o hardware por intermédio de sistema operativo *host*.

[30] Conjunto de políticas ou procedimentos que permitem recuperar documentos em caso de disastre



Fonte : Navigating the IBM cloud, Part 1: A primer on cloud technologies

Cloud Computing

1. Referêncial Teórico

Num cenário de *cloud computing* [31] o utilizador não está preocupado sobre onde estão os seus servidores virtuais ou armazenamento, ele só se interessa se consegue aceder rapidamente às suas fontes quando as necessitar (Tulloch, p. 431), seja para desenvolver uma aplicações ou para testes requerendo hardware. Os programadores podem fazer uso das suas ideias inovadoras para novos serviços de internet, e não necessitam mais de efetuar enormes investimentos em hardware para implementar os seus serviços, ou em custos de mão-de-obra para garantir o seu funcionamento. (Wang, 2011).

Segundo Tim O'Reilly, a computação em *Cloud* é um dos alicerces da nova geração da computação. É um mundo onde as redes são uma plataforma para toda a

[31] É um modelo para habilitar ubíquo, conveniente, a pedido acesso a fontes de computadores configuravéis

computação, onde tudo o que pensamos que é um computador é apenas um dispositivo que permite a ligação para um computador maior que estamos a construir. A computação em *Cloud* apresenta-se como uma grande forma de pensar no que será possível distribuir ao consumidor final em termos de serviços de computação no futuro. (O'Reilly, 2014)

Cloud é um servidor virtual usado para fornecer diferentes perfis de serviço a pedido do utilizador. A palavra *Cloud* é nos dias de hoje a palavra mais usada no mundo de tecnologias de informação e é um modelo que disponibiliza ubíquo, conveniente, a pedido, acesso a um conjunto de recursos de máquinas configuráveis como internet, servidores ou armazenamento que podem ser aprovisionadas ou não com o mínimo esforço do gestor ou interação do provedor (Santana, p. 895). *Provedor* esse que faz parte dos cinco atores de uma *cloud computing* que é uma organização ou entidade responsável pela disponibilização do serviço para o *Cloud Consumer*.

2. As características de uma *Cloud*

A pedido: Onde o consumidor pode aprovisionar unilateralmente capacidades da computação, como tempo do servidor, ou capacidade de internet de uma forma a não necessitar de interação humana. (Technology)

Acesso à rede: Recursos estão disponíveis através de mecanismos standard sejam cliente finos ou não pelas diferentes plataformas como portáteis ou telemóveis. (Technology)

Pool de recursos: recursos de computação do provedor são reunidos para servir vários consumidores usando um modelo multi-tenant, com diferentes recursos físicos e virtuais atribuídos dinamicamente e atribuídos de acordo com a demanda do consumidor. Há um senso de independência localização em que o assinante geralmente não tem controle ou conhecimento sobre a localização exata dos recursos disponibilizados, mas pode ser capaz de especificar o local em um nível mais alto de abstração (por exemplo,

país, estado, ou centro de dados). Exemplos de recursos incluem o armazenamento, processamento, memória, largura de banda de rede e máquinas virtuais. (Technology)

Elasticidade rápida: capacidades podem ser rápida e elasticamente provisionadas, em alguns casos, automaticamente, para escalar rapidamente para fora e rapidamente libertado para escalar rapidamente para o consumidor, os recursos disponíveis para realizar o provisionamento muitas vezes parecem ser ilimitadas e podem ser adquiridos em qualquer quantidade a qualquer momento. (Technology)

Serviço medido: sistemas de nuvem podem controlar e otimizar a utilização de recursos, aproveitando uma capacidade de medição em algum nível de abstração apropriado para o tipo de serviço (contas por exemplo, armazenamento, processamento, largura de banda, e usuário ativo) automaticamente. O uso de recursos pode ser monitorado, controlado e reportado, oferecendo transparência tanto para o provedor e consumidor do serviço utilizado. (Technology)

3. Os Modelos de Serviço

Software como um serviço (*SaaS*) : As aplicações são acedidas de vários dispositivos de clientes através de um interface de um cliente fino como um Web browser, o consumidor não gere ou controla a infraestrutura incluindo internet, servidores ou armazenamento. (Technology)

Plataforma como um serviço (*PaaS*): A capacidade fornecida ao consumidor é implantar para as aplicações de infraestrutura de nuvem consumidor-criadas ou adquiridas criados usando linguagens de programação e ferramentas suportadas pelo provedor. O consumidor não gerir ou controlar a infra-estrutura de nuvem subjacente, incluindo rede, servidores, sistemas operacionais, ou armazenamento, mas tem o controle sobre os aplicativos implantados e configurações de ambiente, possivelmente de aplicação de hospedagem. (Technology)

Infraestrutura como um serviço (*IaaS*): A capacidade fornecida ao consumidor é o de processamento de fornecimento, armazenamento, redes e outros recursos computacionais fundamentais em que o consumidor é capaz de implantar e executar software arbitrário, que pode incluir sistemas operacionais e aplicativos. O consumidor não gerir ou controlar a infra-estrutura de nuvem subjacente, mas tem o controle sobre os sistemas operacionais, armazenamento, aplicativos implementados e controle, possivelmente limitado de componentes de rede selecionados. (Technology)

4. Os Atores de uma *Cloud*

São cinco os atores referentes a todo o desempenho de uma *cloud* desde o *consumer*, que pode ser uma organização que mantem uma relação de trabalho e usa serviço do *provider*,mas também pode pedir serviços a um *cloud provider,* este sim é uma organização responsável por fazer um serviço disponível para as partes interessadas, serviço que é controlado por um *auditor* que conduz audições independentes e pode contactar os outros para colecionar informação necessária. O ator *broker* é a entidade que gera o uso, performance e entrega de serviços e negoceia relações entre o *provider* e o *consumer* mas nada funcionava sem o *carrier* que providencia a conectividade e transporte dos serviços *cloud* do *provider* para o *consumer*. (Technology)

Trabalho Digital

A aplicação Trabalho 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 utilizador da Biblioteca tem credenciais próprias, nome de utilizador e palavra passe, para ter acesso remoto a uma máquina virtual.

Após ter iniciado sessão remota na sua máquina a aplicação Trabalho Digital inicia automaticamente apresentando uma janela onde o utilizador deverá introduzir as suas credenciais de acesso à aplicação.

De seguida poderá efetuar uma pesquisa pela base de dados, consultas de livros, ou link's de internet, para desenvolver o seu conhecimento.

O utilizador terá ao seu dispor uma cloud da Microsoft que servirá de apoio aos seus documentos que tenham sido préviamente guardados ou consultados.

A aplicação foi desenvolvida utilizando o software de programação Microsoft Visual Studio 2013.

Criação das Máquinas Virtuais

1. Arranque por Disco Virtual VHD

Em primeiro, e para um melhor desempenho optámos por um arranque de SO Windows Server 2012 através de um VHD para se obter uma melhor performance.

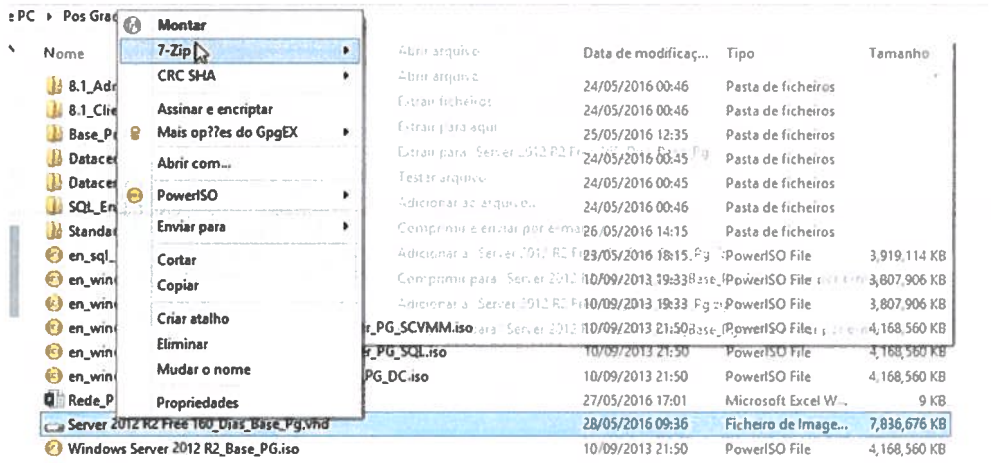


Figura 1 - Montagem Disco VHD

Depois de obtido com sucesso a montagem da máquina VHD, definimos através do comando MSCONFIG o boot opcional de arranque.

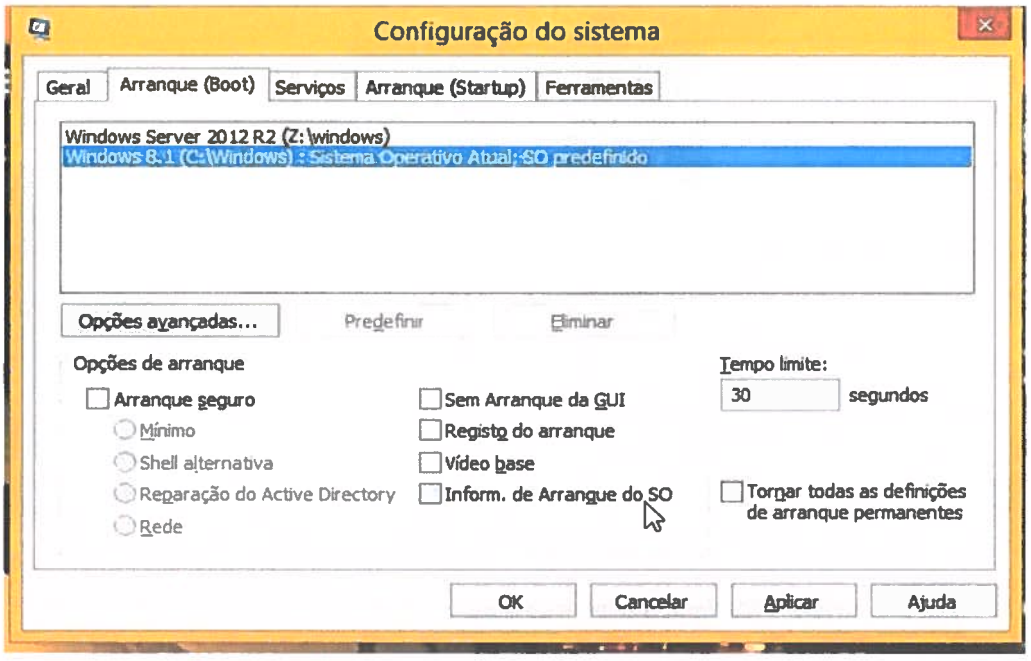


Figura 2 - Opção de Arranque do SO

2. Instalação das Máquinas Virtuais no *VMware Workstation 12*

O *hypervisor* da *Vmware* foi o escolhido para configuração das máquinas virtuais, DC1, VMM, SQLServer e máquinas cliente, para efectuarmos as suas instalações clicamos na consola em *Create a new Virtual Machine.*

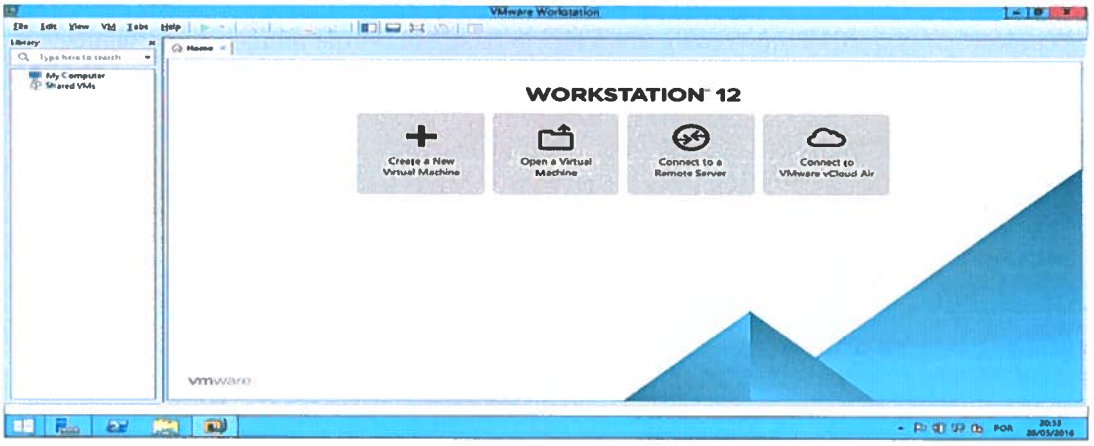


Figura 3 - Consola VMware Workstation

No *Wizard* que aparece de seguida optamos pela instalação *Custom (Advanced)*, por se definir melhor a estrutura das máquinas virtuais.

Figura 4 - Criação das máquinas virtuais com opções avançadas

No processo de configuração das máquinas virtuais, verifica-se no *Wizard* seguinte a compatibilidade em relação ao *Workstation.*

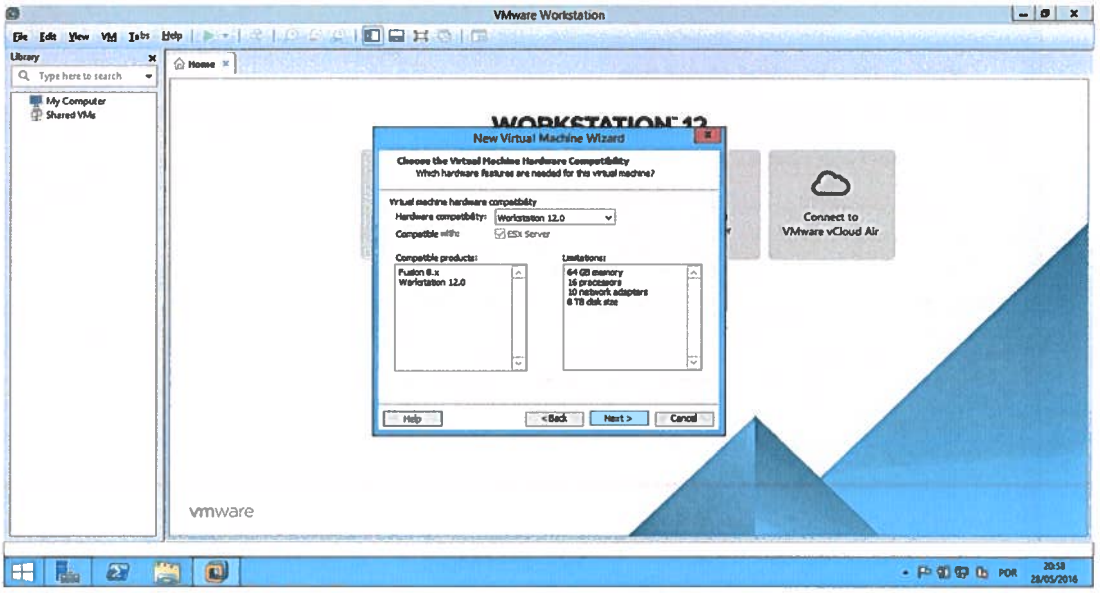


Figura 5 - Compatibilidade Virtual Machine

Aqui para se procurar o caminho do Sistema Operativo em formato *iso*[32] que vai servir de Controlador de Domínio clicamos no *browse.*

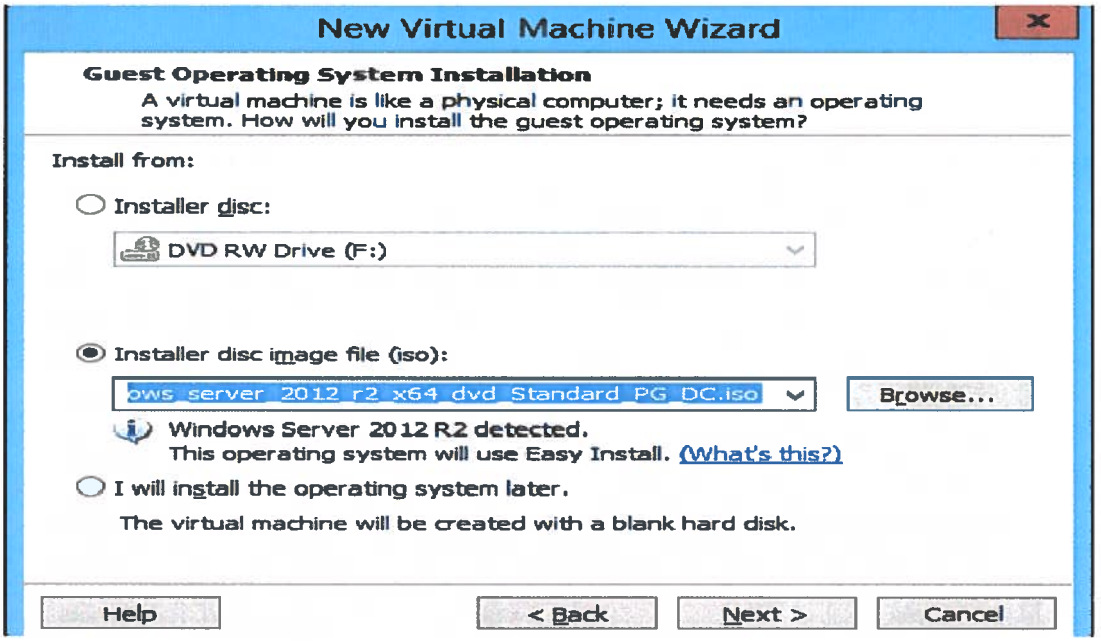


Figura 6 - Caminho ISO para instalação do DC1

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

Para o desenvolvimento do Projeto, todo o processo iria decorrer num Domínio, pelo que optámos por designar o nome de DC1 para instalação do mesmo.

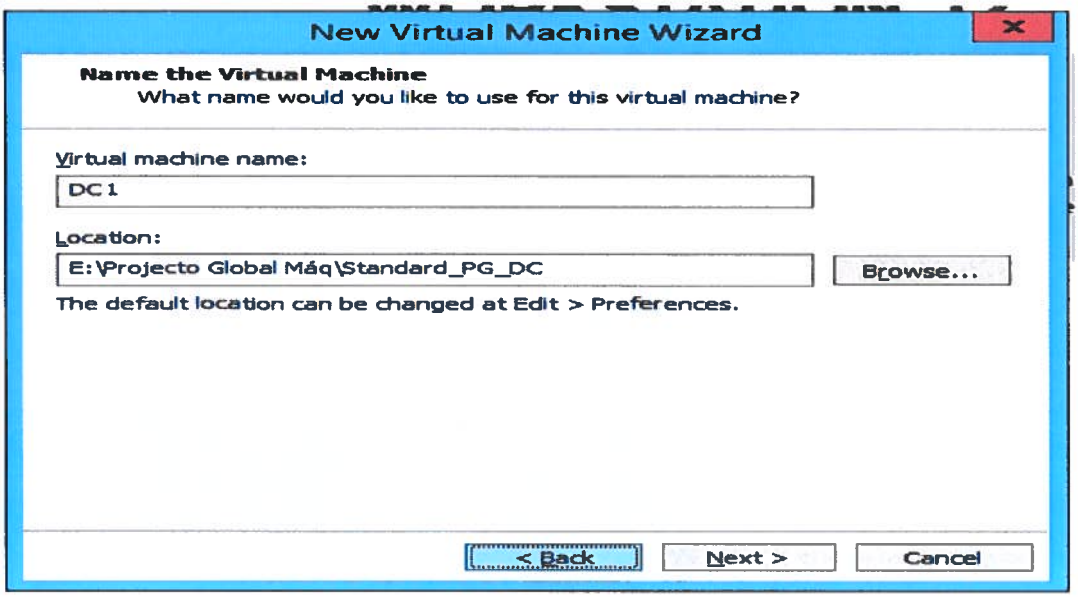


Figura 7 - Nome da Máquina Controlador de Domínio

Designamos a capacidade de 1024 MB de memória RAM[33] visto que o controlador de domínio só iria alojar as máquinas, não sendo necessário mais para este efeito.

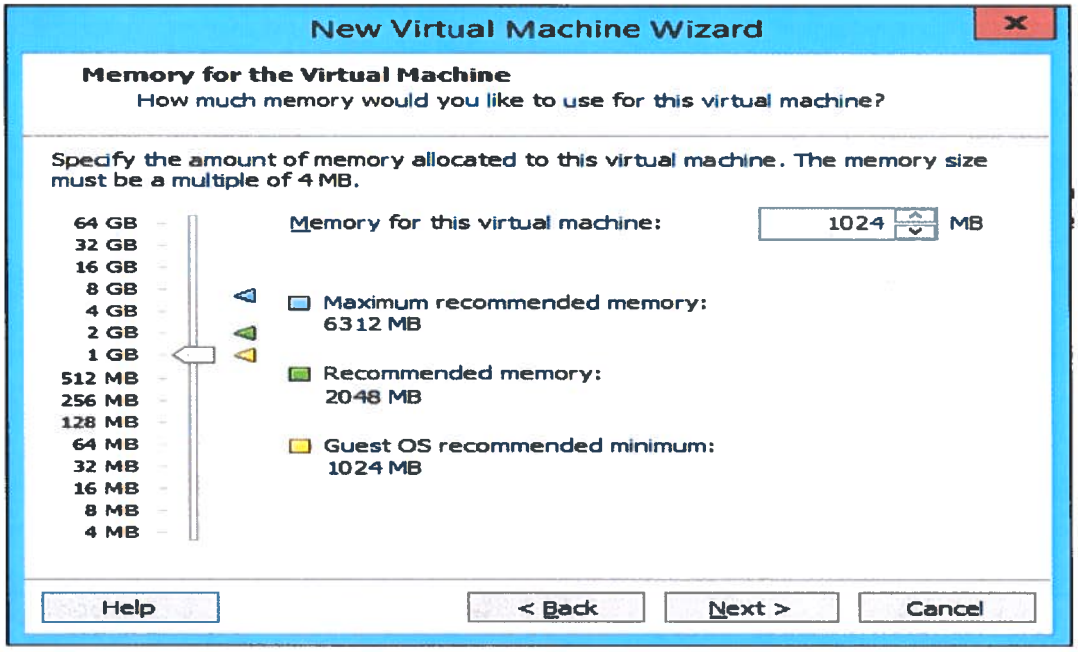


Figura 8 - Capacidade de memória RAM no Controlador de Domínio

[33] Random Access Memory

Para se obter acesso à internet através da placa de rede escolhemos a NAT, que vai comunicar com a placa do host, esta opção foi para se obter as atualizações do sistema operativo.

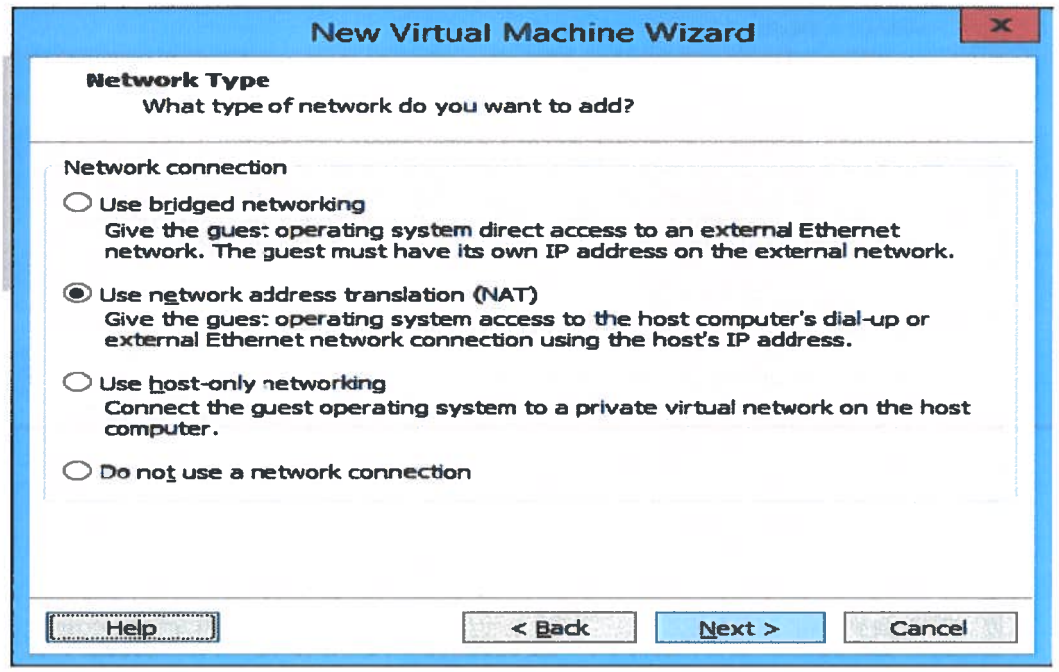


Figura 9 - Escolha Placa de Rede

Na definição do disco optamos um disco scsi[34]

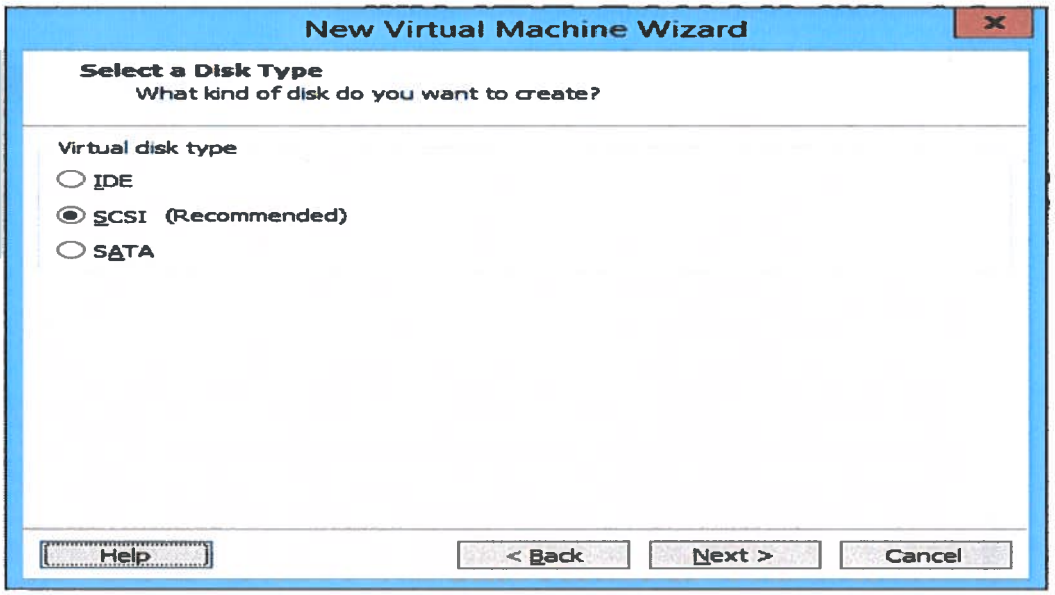


Figura 10 - Tipo de Disco

[34] Small Computer System Interface

O disco na sua criação é novo, dado que o laboratório vai ser inicializado a partir do zero.

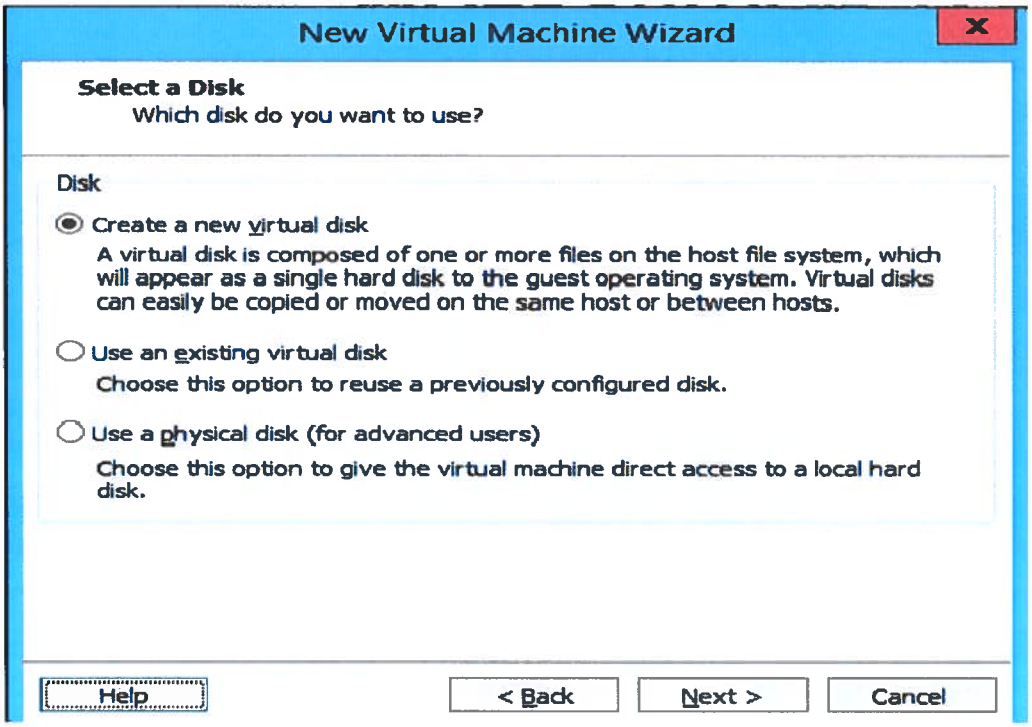


Figura 11 - Criação do Disco

Para finalizar, definimos a pasta onde vai ficar todo o processo de instalação da máquina controlador de domínio anteriormente comentado.

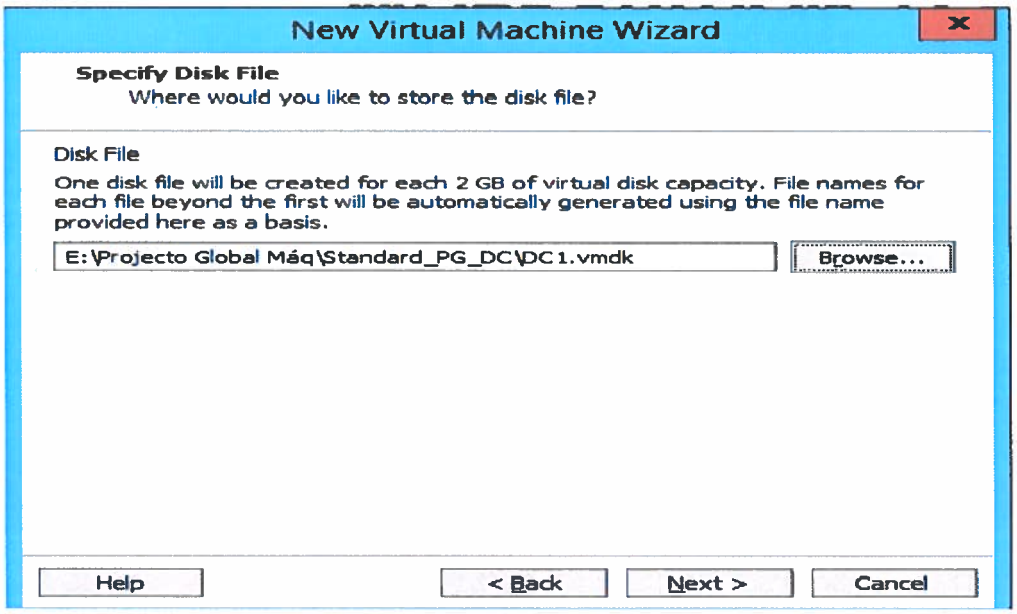


Figura 12 - Caminho onde o Disco fica alojado

3. Instalação da Máquina que vai servir de Controlador de Domínio

Dado a necessidade do laboratório que vamos efectuar iremos proceder às configurações do mesmo a fim de fazer os pré-requesitos para instalação do DC1 (ipfixo, atualizações, atribuição de um nome à máquina).

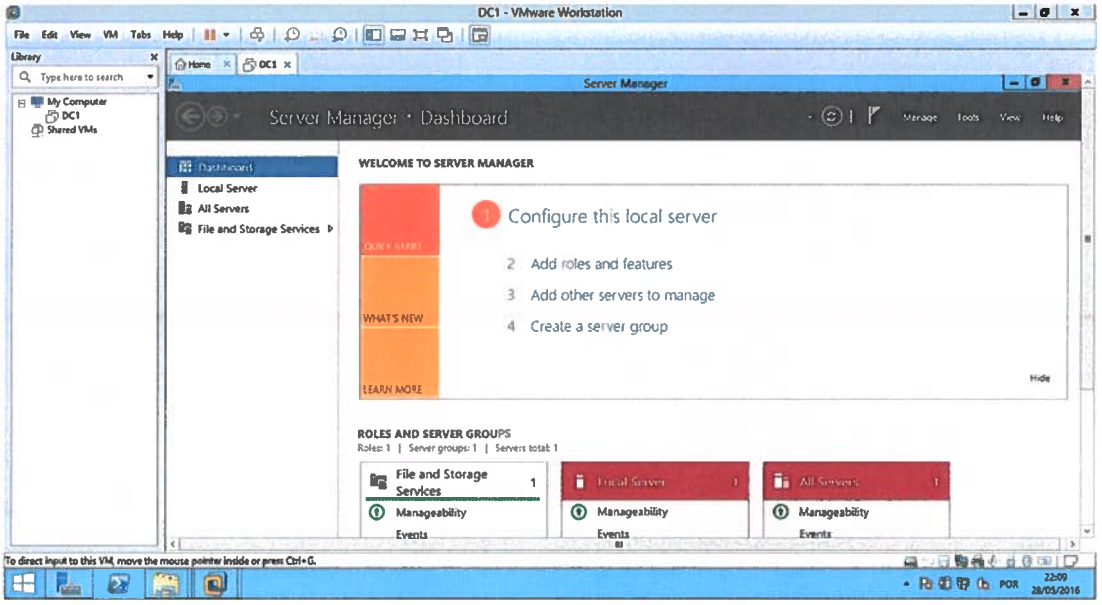


Figura 13 - Consola Inicial do Controlador de Domínio

Aqui verifica-se o primeiro requesito obrigatório para a instalação de um controlador de domínio que é a atribuição de um nome à máquina.

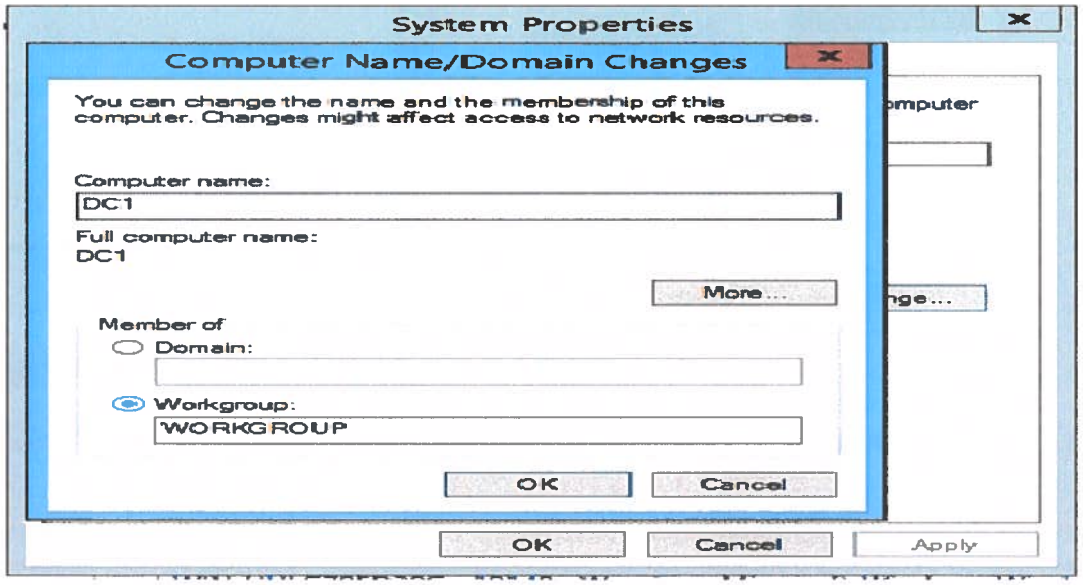


Figura 14 – Atribuição do Nome da Máquina para Controlador de Domínio

O segundo requesito aqui demonstrado são as placas de rede (host only, nat), obrigatóriamente teria de existir uma placa com um ip fixo para servir de comunicação entre as máquinas que nos processos seguintes vão fazer parte deste projecto.

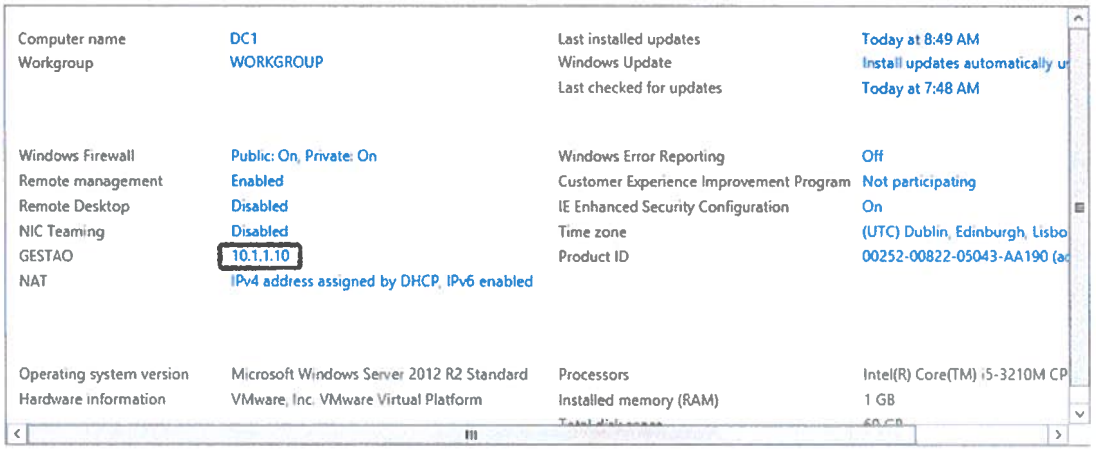


Figura 15 - Atribuição do IP Fixo

A Role necessária para se obter um Controlador de Domínio é a ADDS (active directory domains services), que permite todo um gerenciamento seguro e centralizado de toda uma rede.

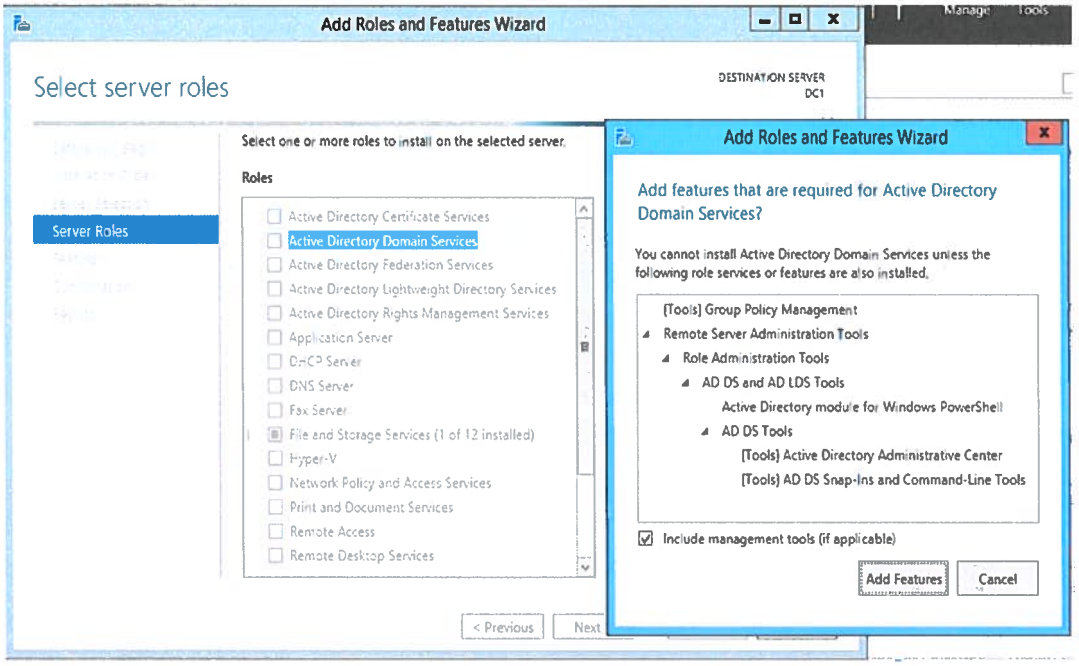


Figura 16 - Instalação da Role ADDS

Por fim vamos promovê-lo a controlador de domínio, neste *wizard* tambem podemos verificar todos os passos dados até aqui no processo de instalação.

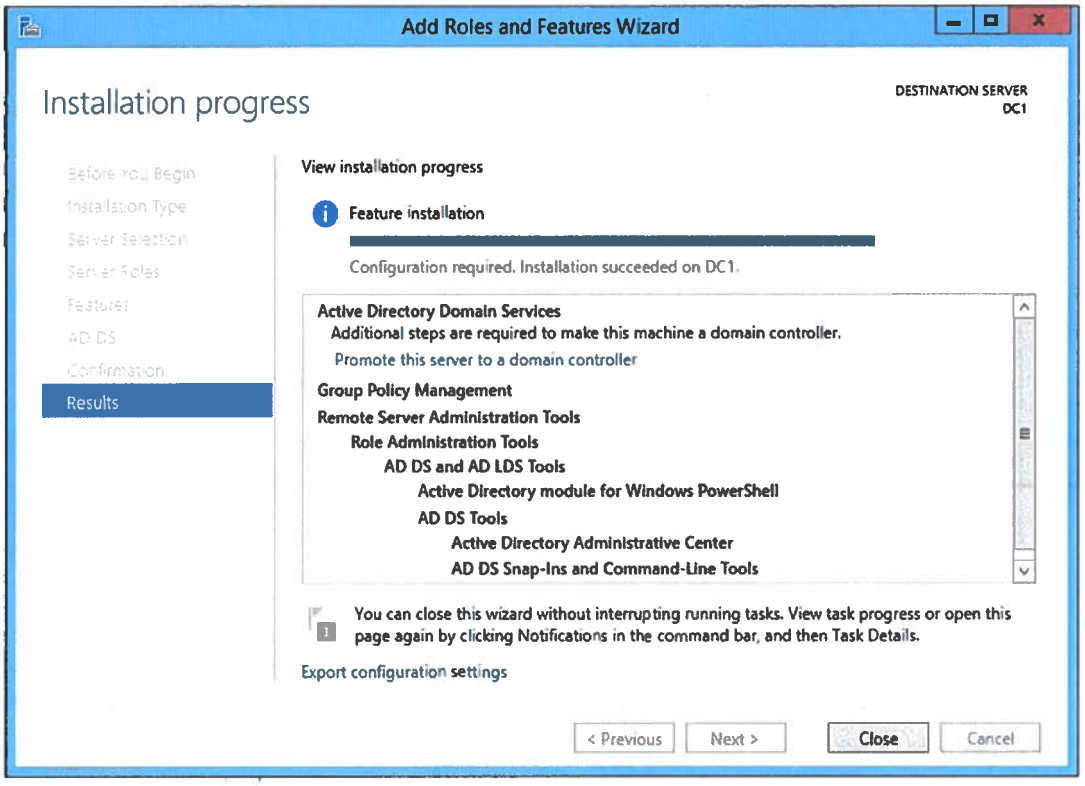


Figura 17 - Promovê-lo a Controlador de Domínio

Tendo efetuado todo um conjunto de pré-requesitos, iremos agora proseguir com a instalação da nossa floresta.

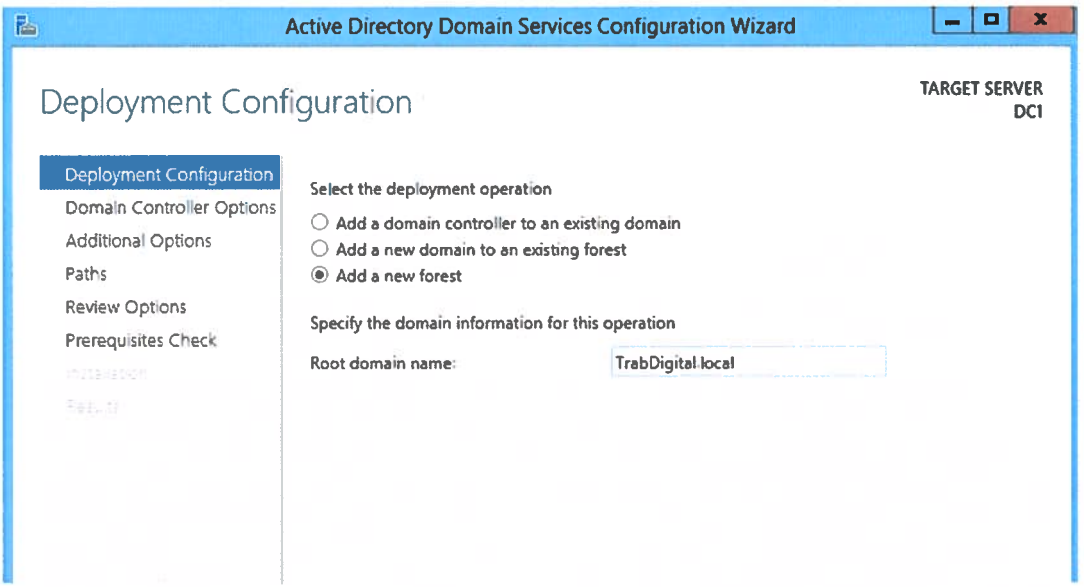


Figura 18 - Atribuição do Nome ao Domínio

Neste *wizard* podemos ver o nível funcional, do nosso domínio e da floresta.

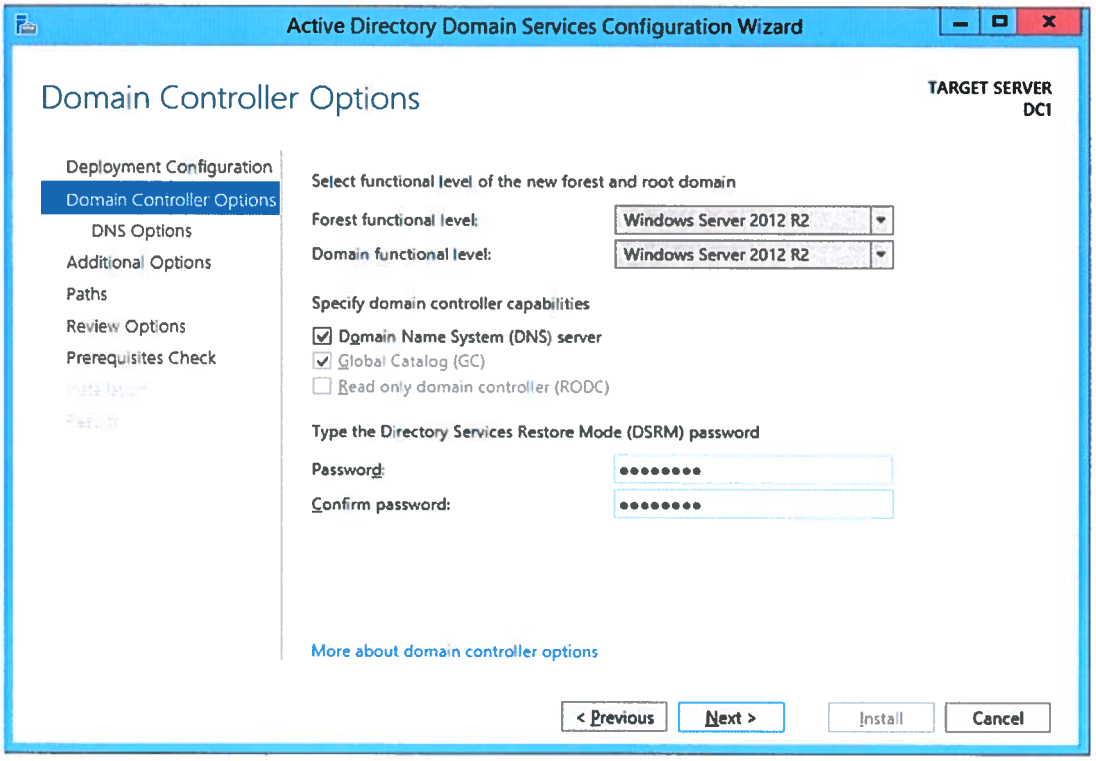


Figura 19 - Nivel Funcional da Floresta e do Domínio

Na continuação do nosso projeto verifica-se aqui o *NetBios* domain name

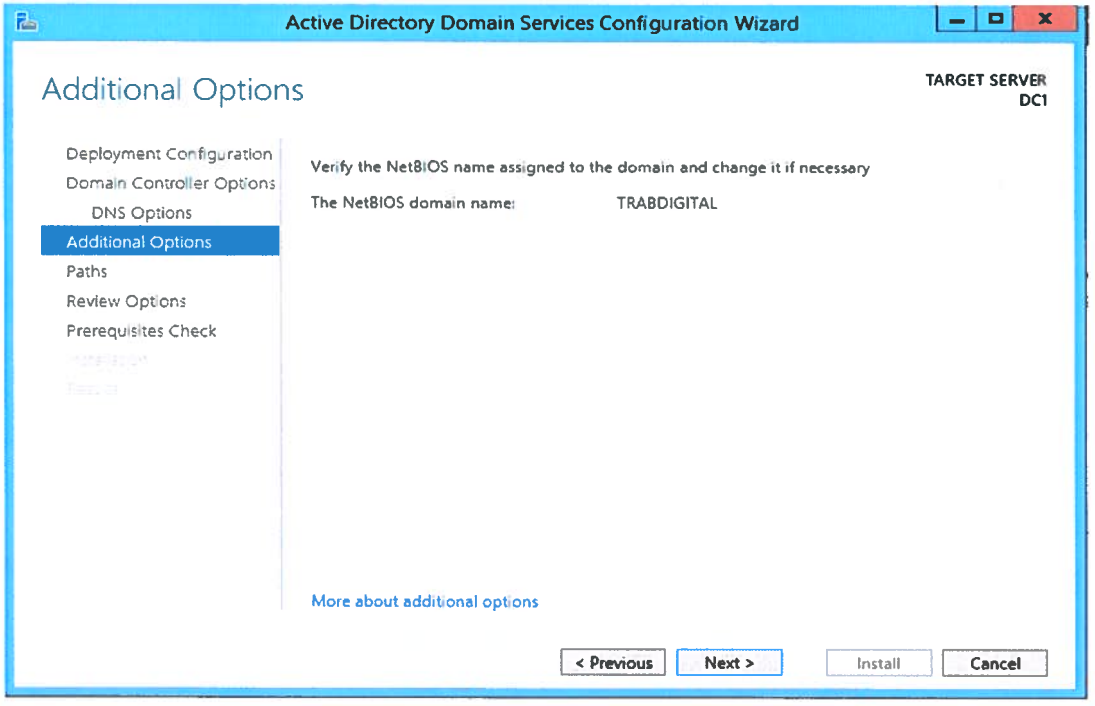


Figura 20 - NetBios Domain Name

Neste momento obtêm-se a localização do SYSVOL, Database e Log Files, a Microsoft recomenda não instalar estes três ficheiros juntos do sistema operativo.

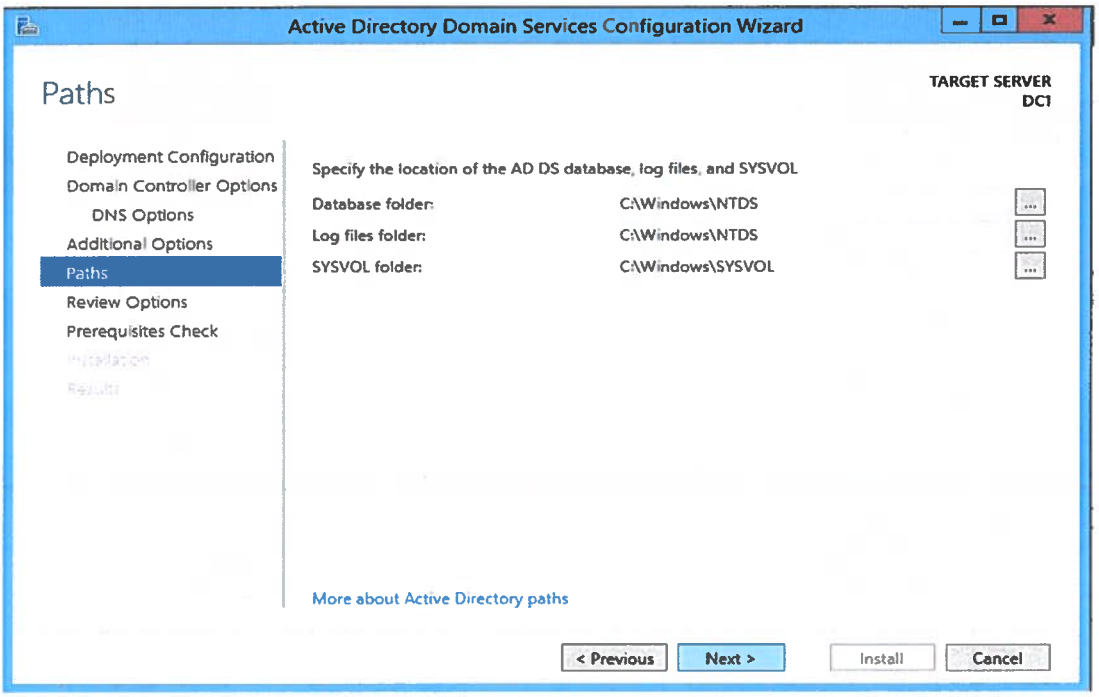


Figura 21 - Localização do SYSVOL, Database, Log Files

Apesar de alguns avisos relacionados com dados criptográficos, temos a validação de todo um processo para obter um controlador de domínio com sucesso.

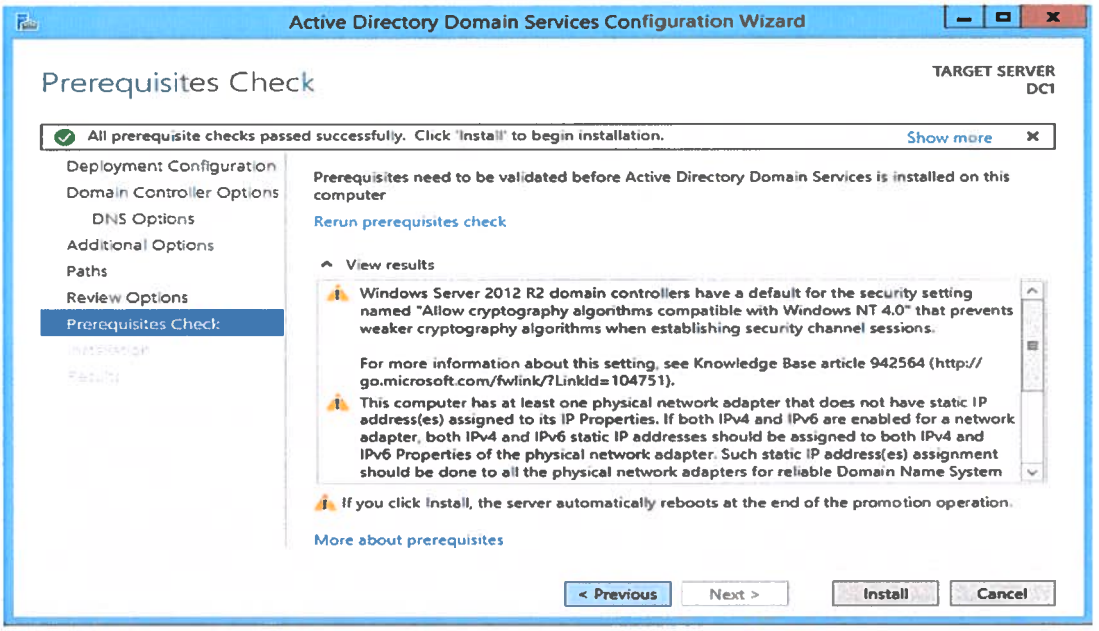


Figura 22 - Verificação de Pré-Requesitos

4. Instalação da Role DHCP Server

No Server Manager, nas roles e features, marca-se a opção DHCP Server, para a instalação da funcionalidade.

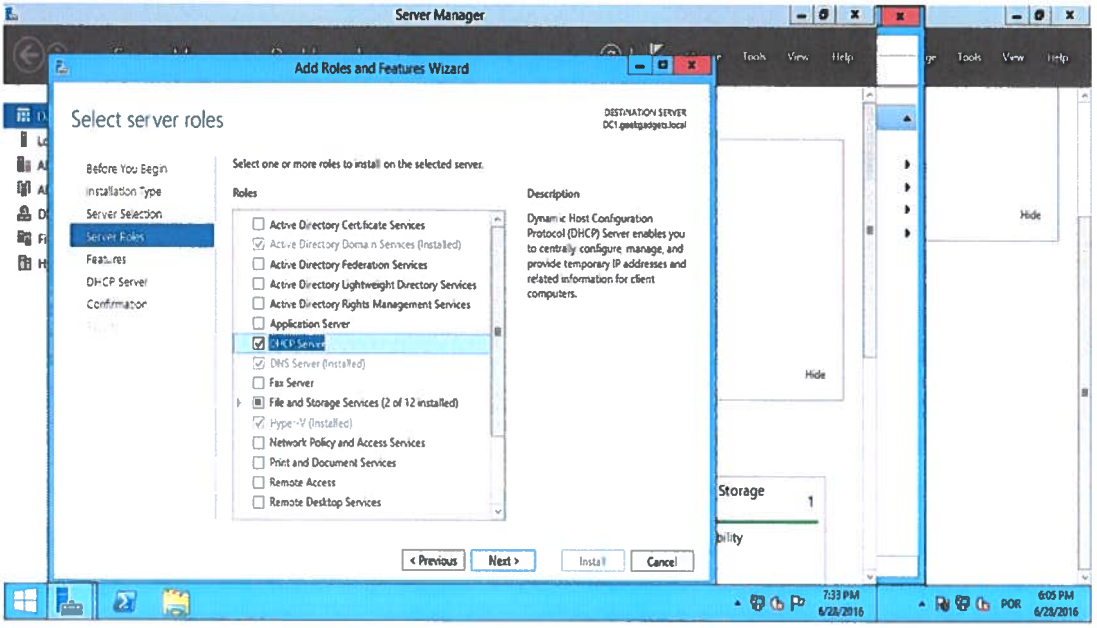


Figura 23 - Instalação DHCP Server

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

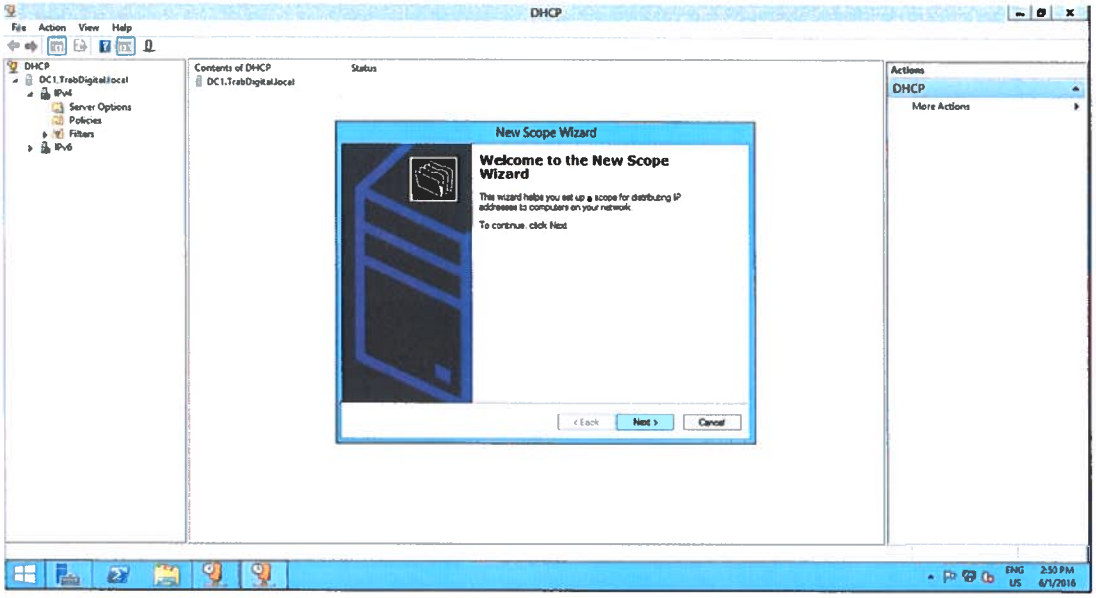


Figura 24 - Consola DHCP

Aqui define-se o nome da scope do DHCP.

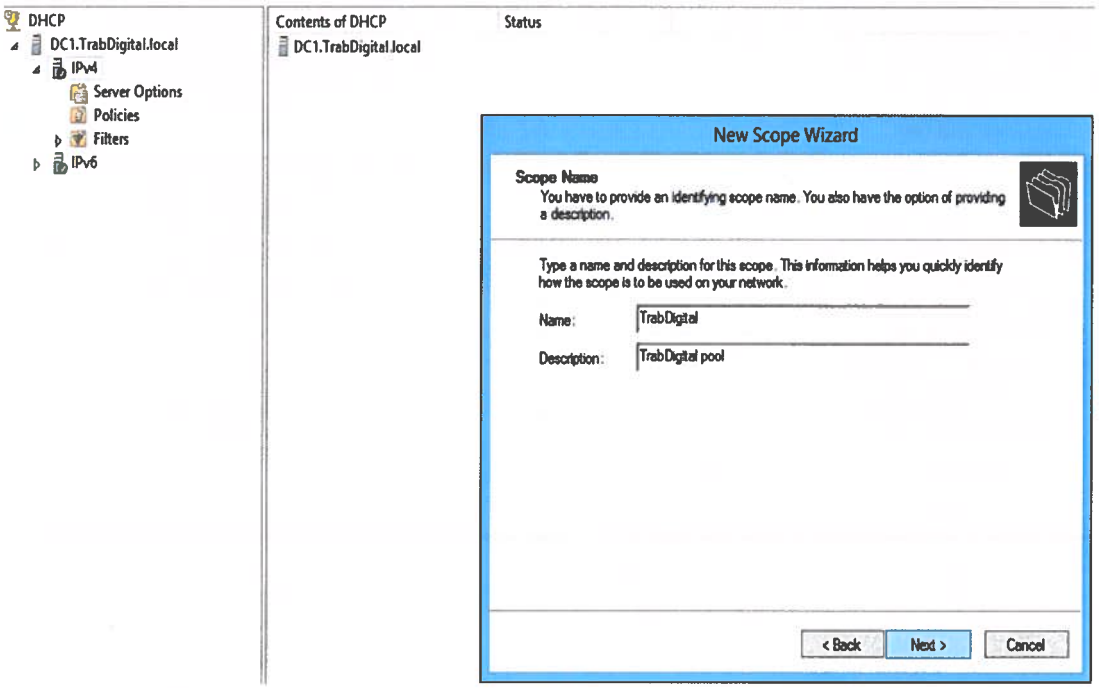


Figura 25 - Nome Scope DHCP

De seguida uma nova range de IP's

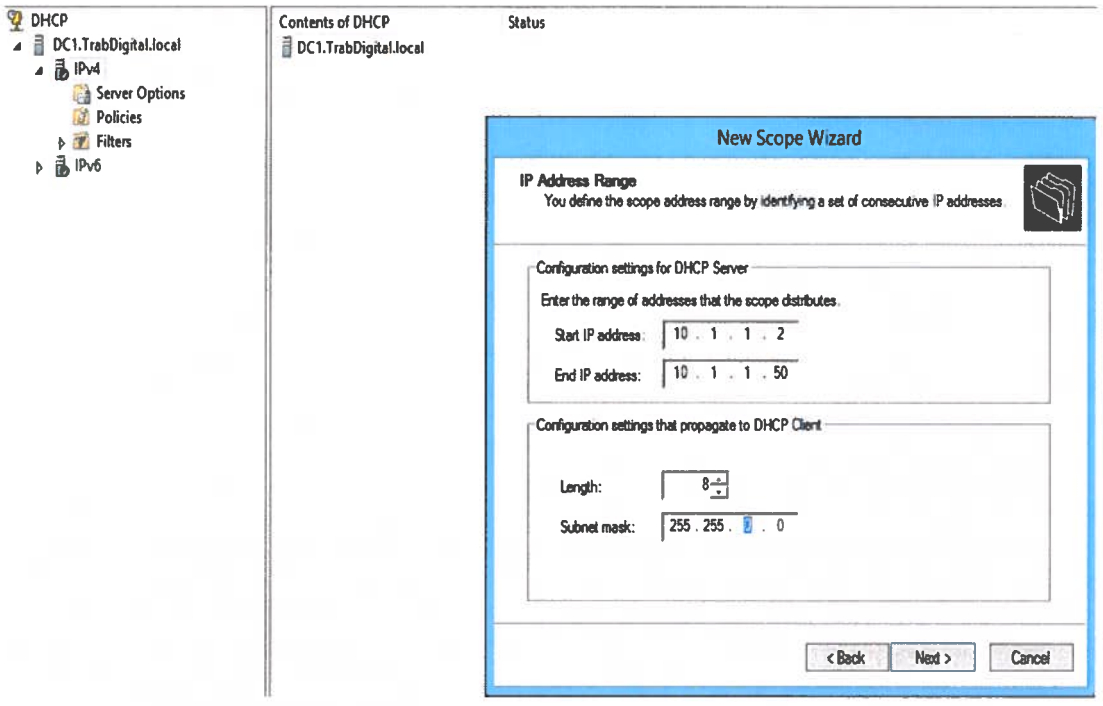


Figura 26 - Atribuição do intervalo de IP's

De seguida em relação á imediata configuração do scope, clicamos no sim.

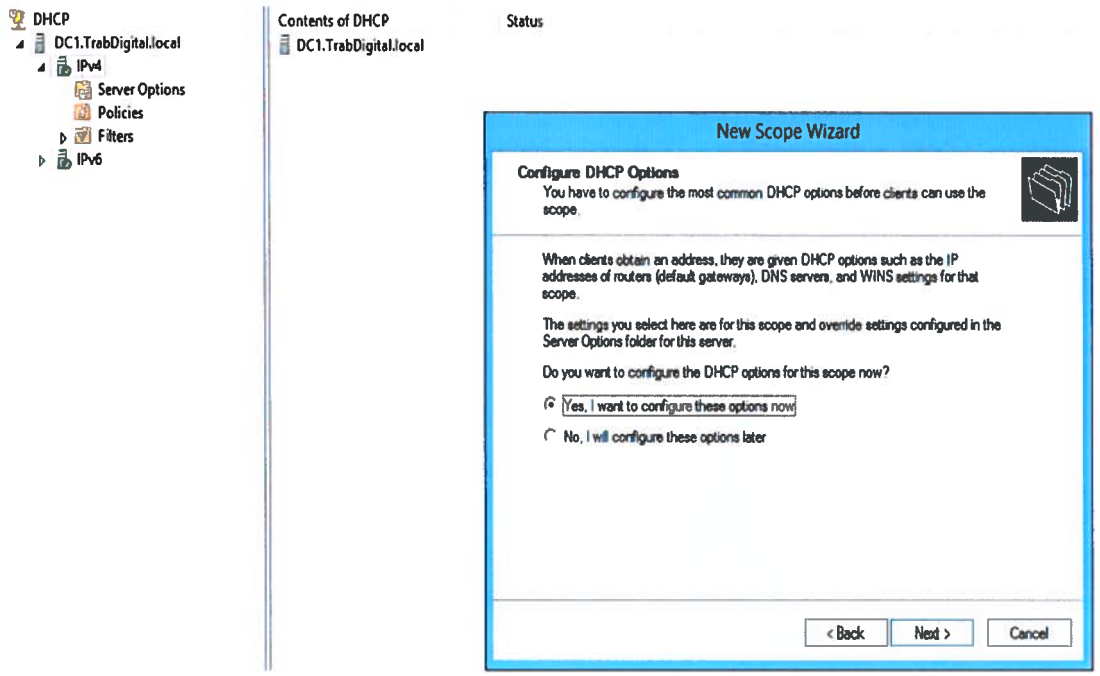


Figura 27 - Início de Configuração de opções do scope

Nas opções do scope especifica-se o *DNS* do Dominio.

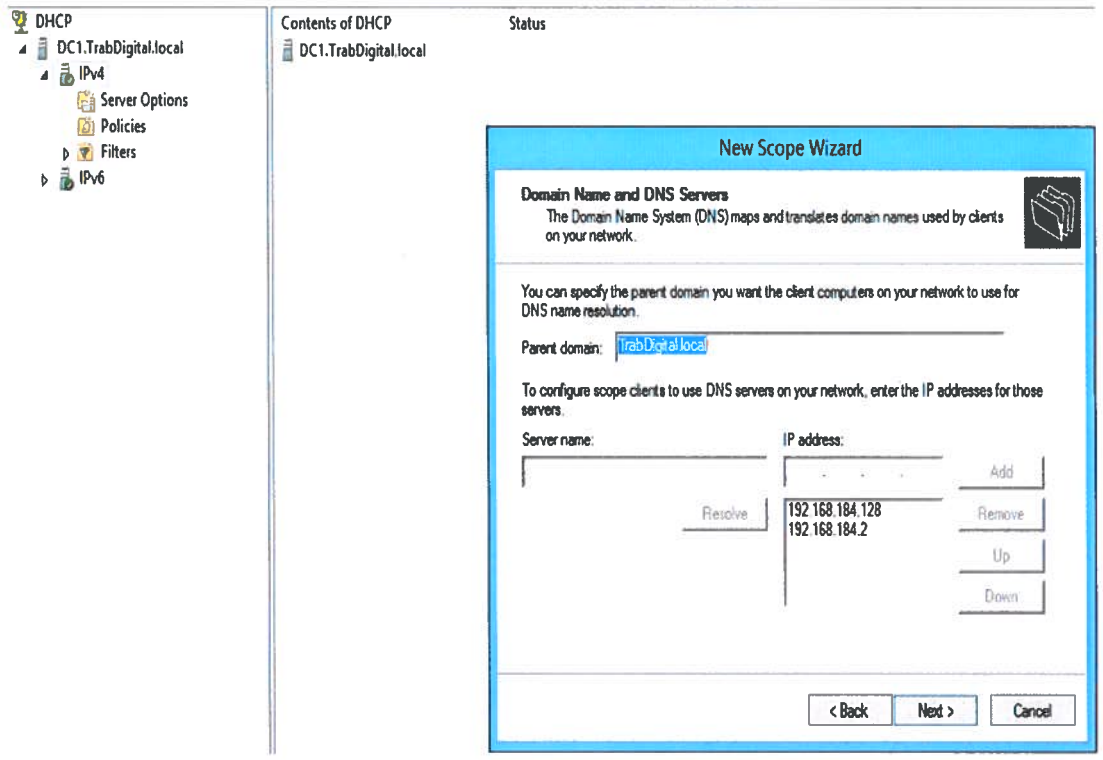


Figura 28 - DNS Server

5. Criação de Unidades Organizacionais e Contas de Serviços

Para permitir que todo este trabalho seja linear, atribuimos um ip, fez-se os updates e iremos agora proseguir com o adicionamento da máquina SQLServer ao Controlador de Domínio.

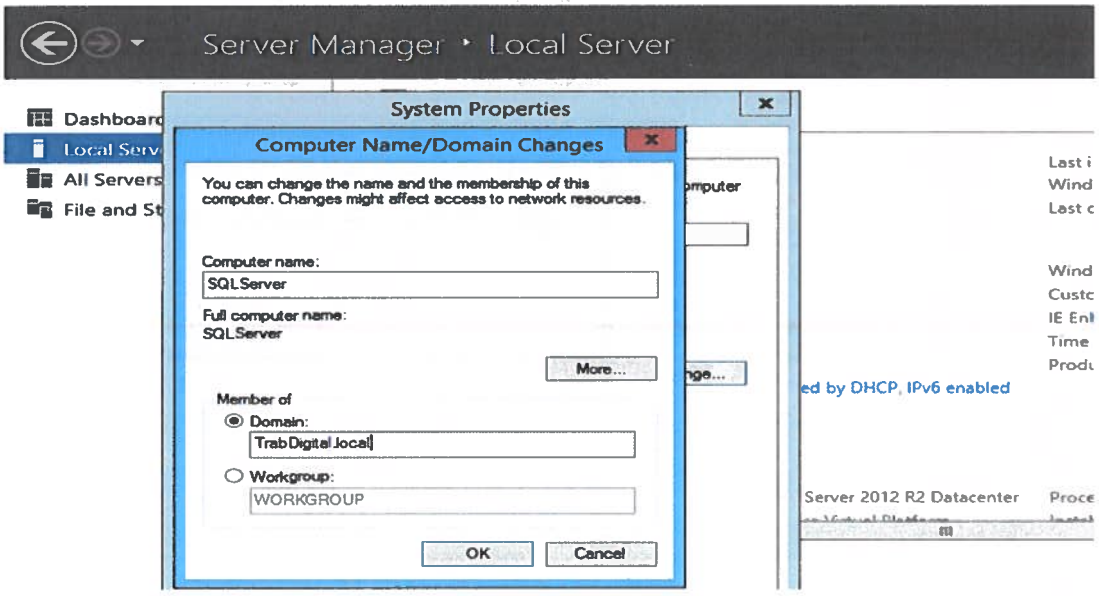


Figura 29 - Atribuição do nome à Máquina SQLServer e adicionar ao Domínio

Obrigatóriamente temos de criar uma Unidade Organizacional no controlador de domínio, e contas de serviços para decorrer no System Center Virtual Machine Manager.

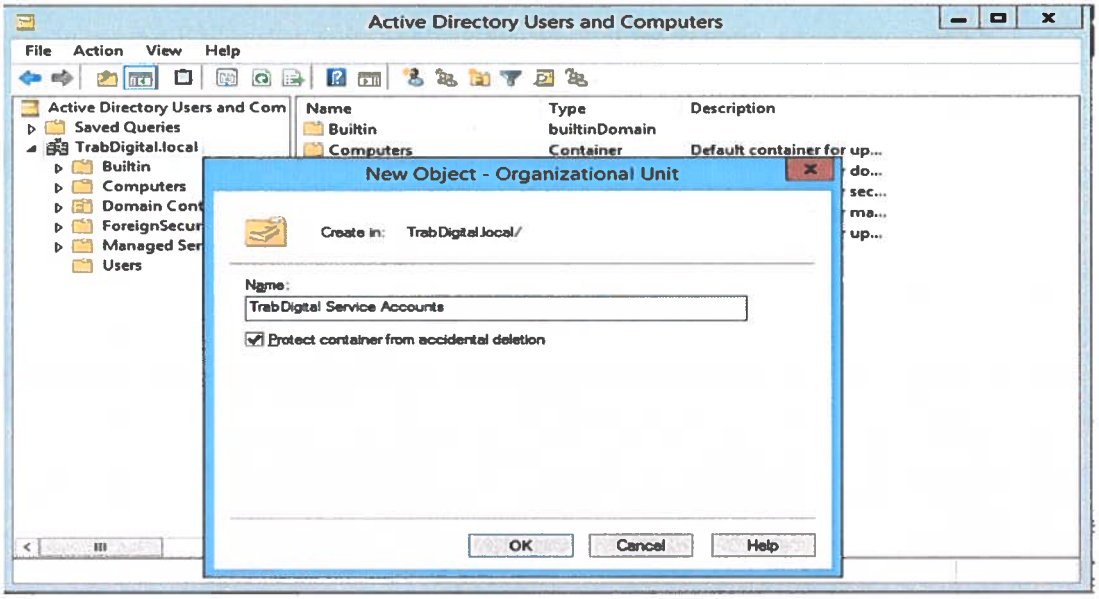


Figura 30 - Criação da unidade organizacional para os serviços do vmm

Aqui verificamos as contas de serviço necessárias para o SQLServer e para o System Center Virtual Machine Manager criadas com sucesso.

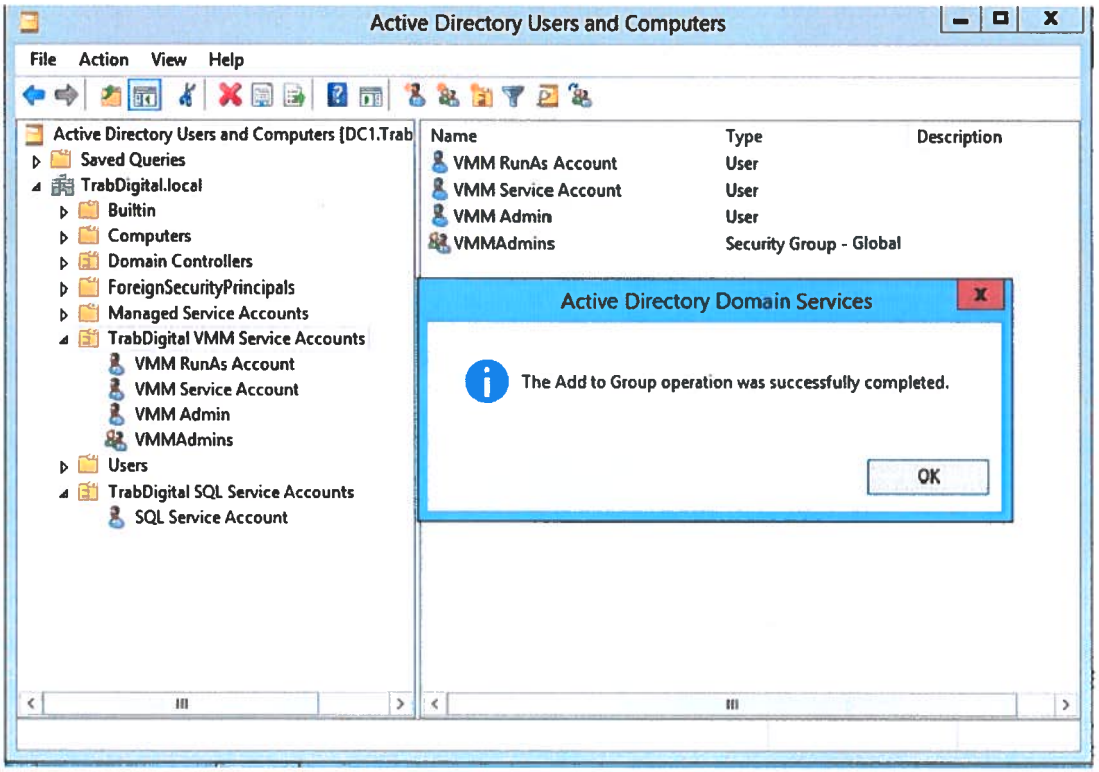


Figura 31 - Contas efetuadas com sucesso para o SQLServer e System Center Virtual Machine Manager

6. Instalação da Máquina SQLServer Enterprise

O próximo passo, após se ter feito as contas de serviço para o SQLServer no Controlador de Domínio será o processo da sua instalação.

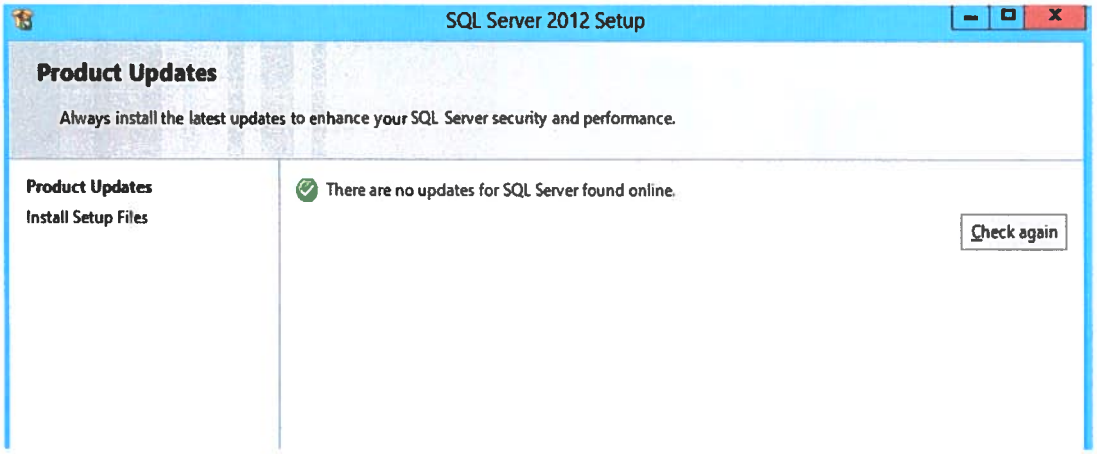


Figura 32 - Updates feitos

Numa primeira abordagem no processo de instalação, deparamo-nos com o método que queremos instalar, que neste caso será o SQL Server Feature Installation.

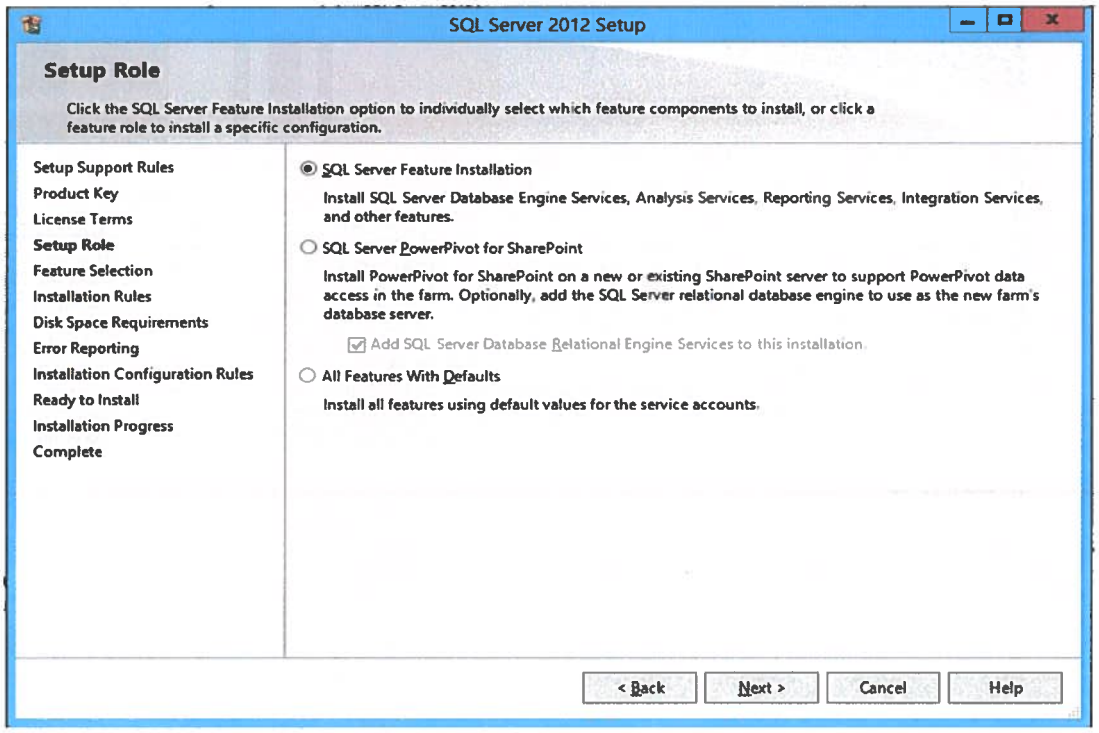


Figura 33 - Role Instalação do SQLServer

Instalação das features necessárias para toda a Biblioteca Virtual.

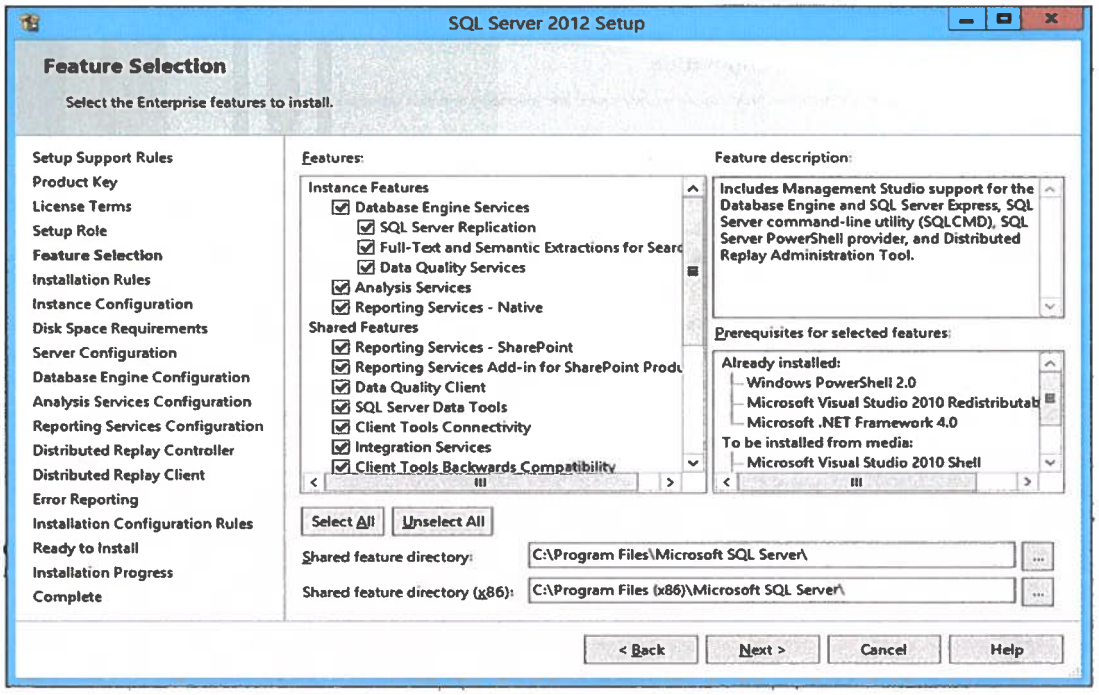


Figura 34 - Instalação das Features

Continuando a instalação, eis uma das partes mais importantes que não poderia ser efetuada de um modo correcto sem que antes se tivessem feito as contas de serviço.

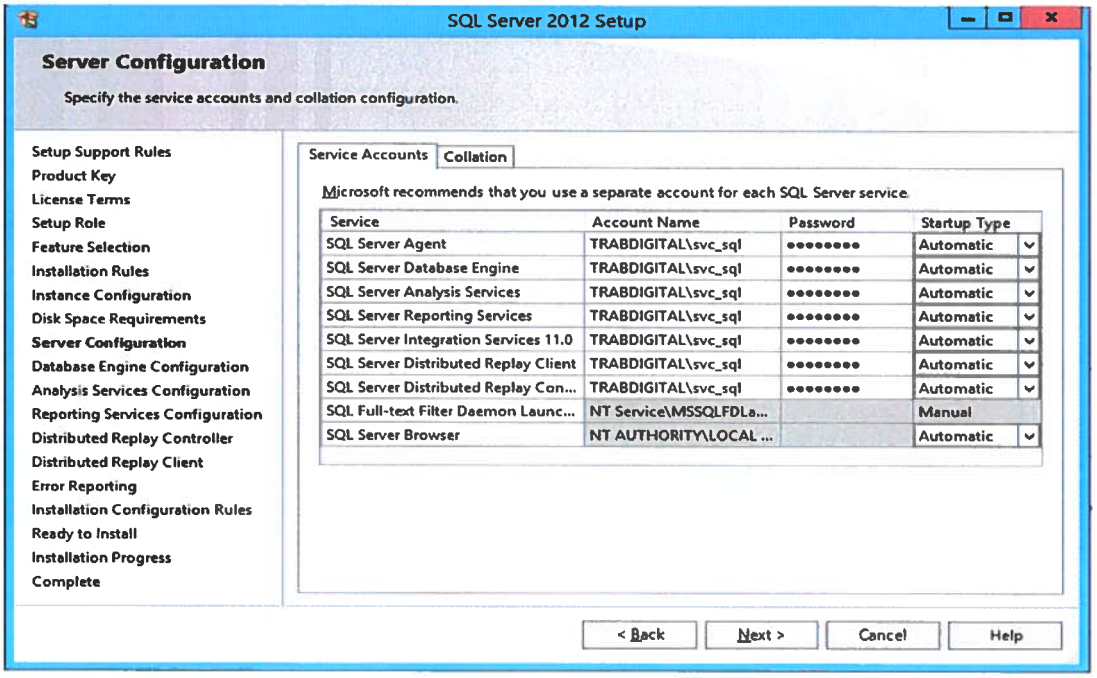


Figura 35 - Configuração do Servidor

Neste *wizard* , especifica-se quem terá privilégios de autenticação de login e o modo como essa autenticação é feita que neste momento será do Windows.

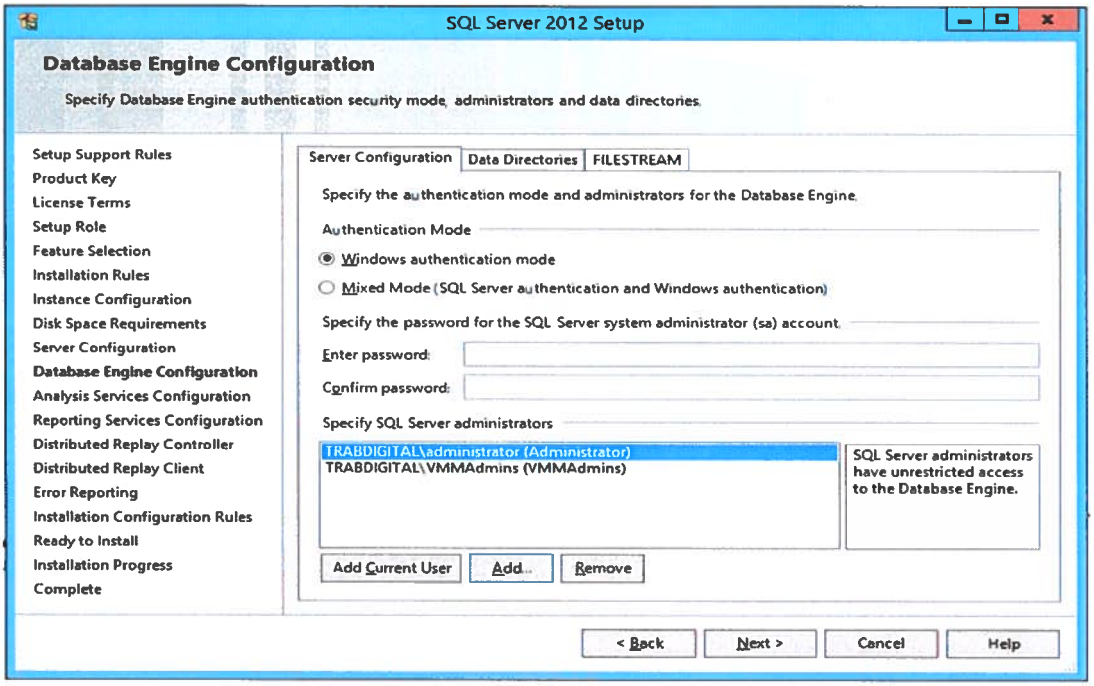


Figura 36 - Administradores da Base de Dados e modo de Autenticação

Depois de analisados os serviços e reportados, reparamos na configuração dos regulamentos para a sua instalação.

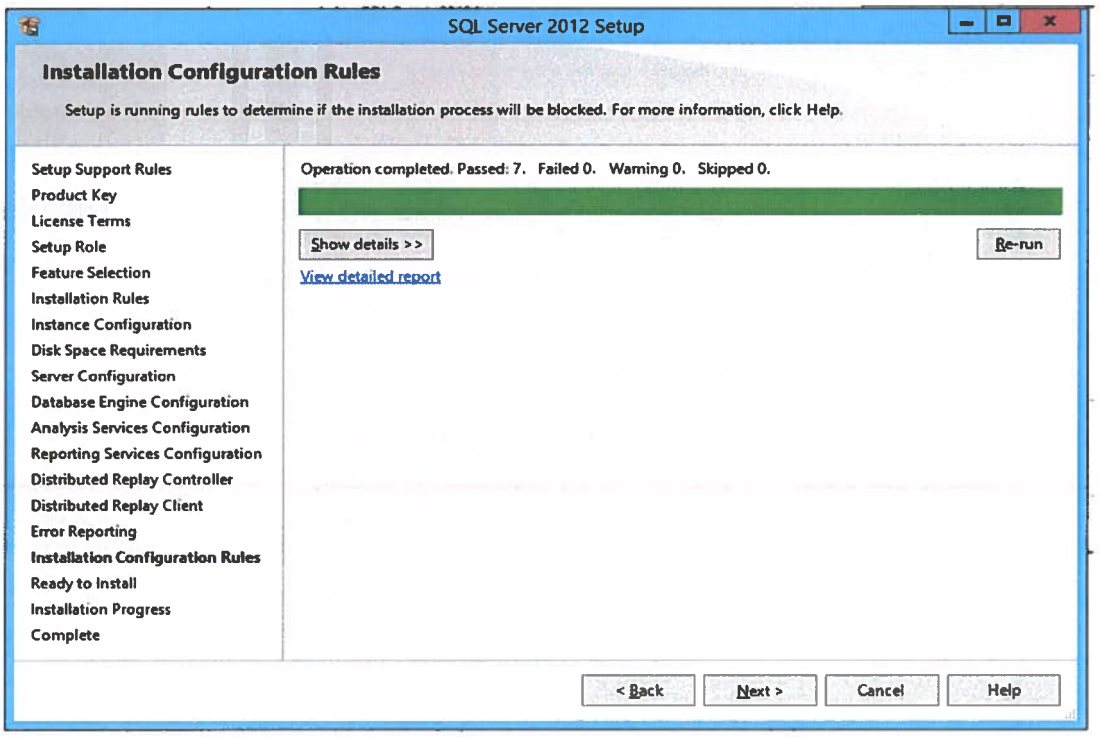


Figura 37 - Configuração

Na parte final, verifica-se todo o processo feito anteriormente.

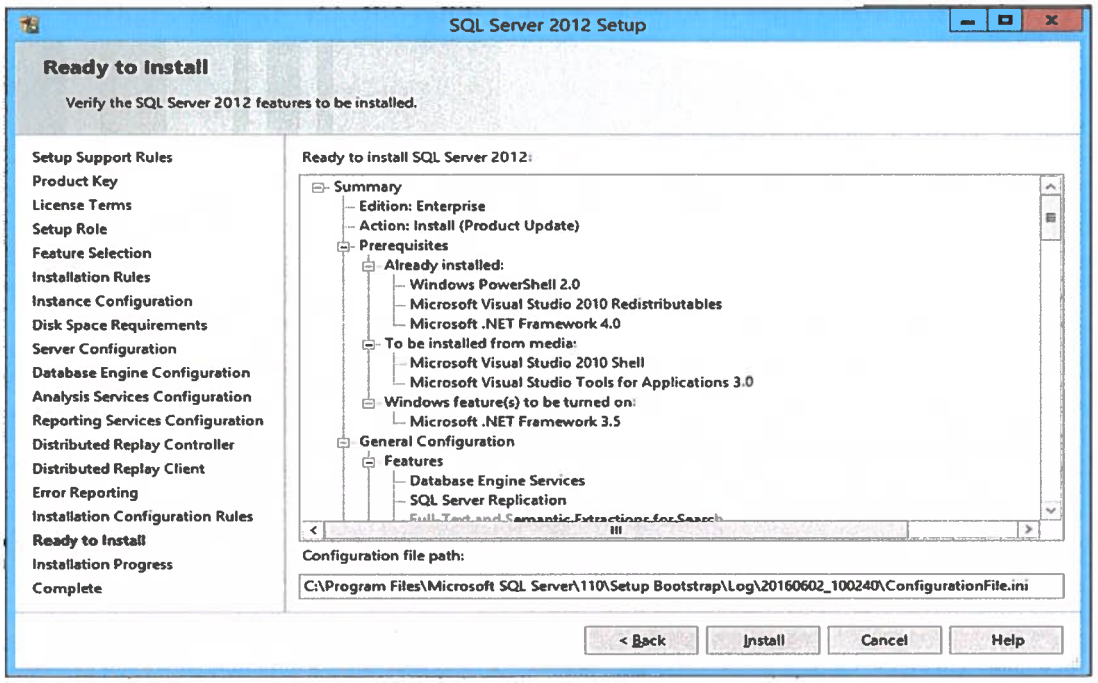


Figura 38 - SQLServer pronto a instalar

No *wizard* final todo o processo criado anteriormente foi feito com sucesso.

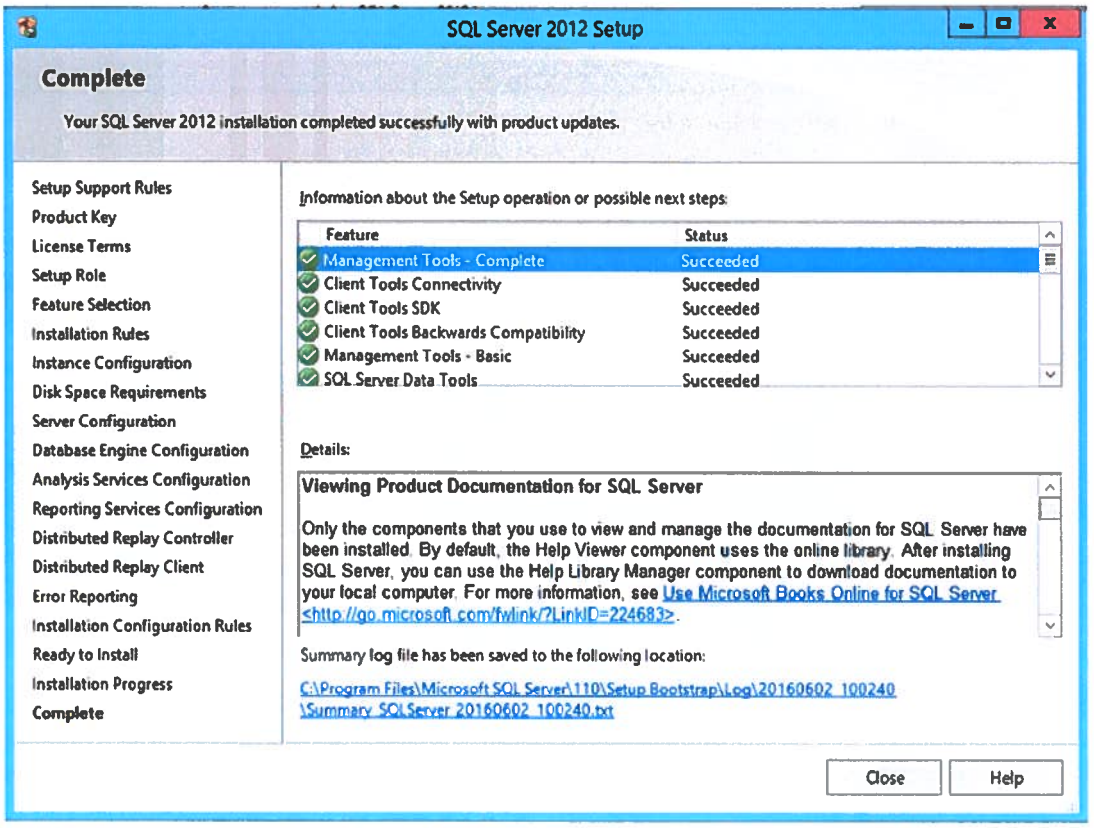


Figura 39 - Instalação Completa

7. Instalação e Configuração do System Center Virtual Machine Manager

Para inicializar a instalação do SCVMM teremos de adiciona-lo ao domínio.

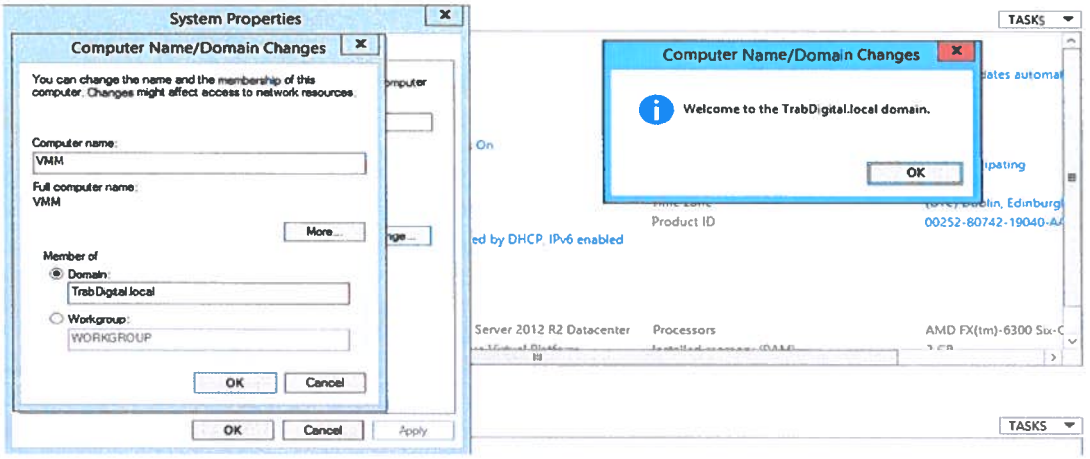


Figura 40 - acesso ao domínio do scvmm

Como foi visualizado anteriormente criamos três contas (administrador, serviços e runas), e um grupo com privilégios de administrador.

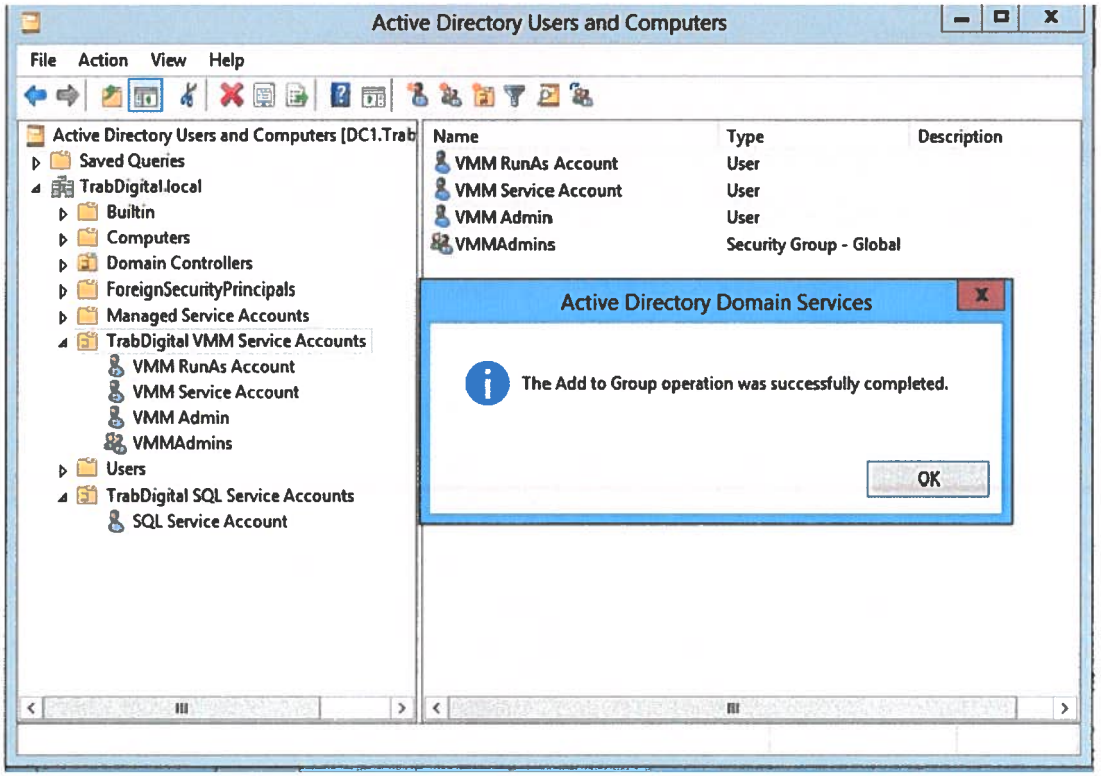


Figura 41 - Contas de Serviços e Administração

Depois de criarmos os grupos necessários, vamos começar com a instalação do SCVMM.

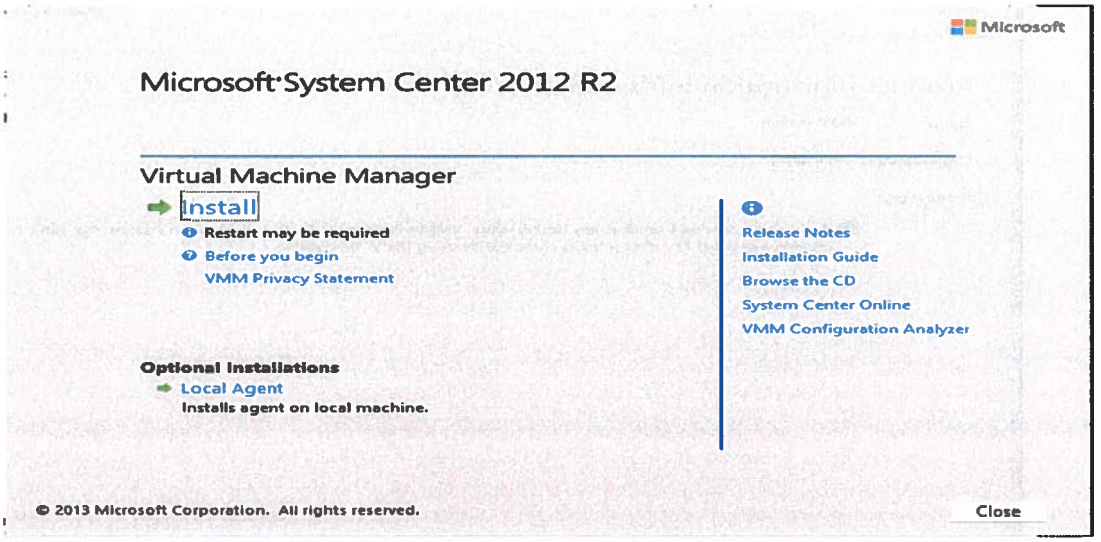


Figura 42 - Consola inicial SCVMM

O segundo *wizard* indica-nos as features que queremos instalar.

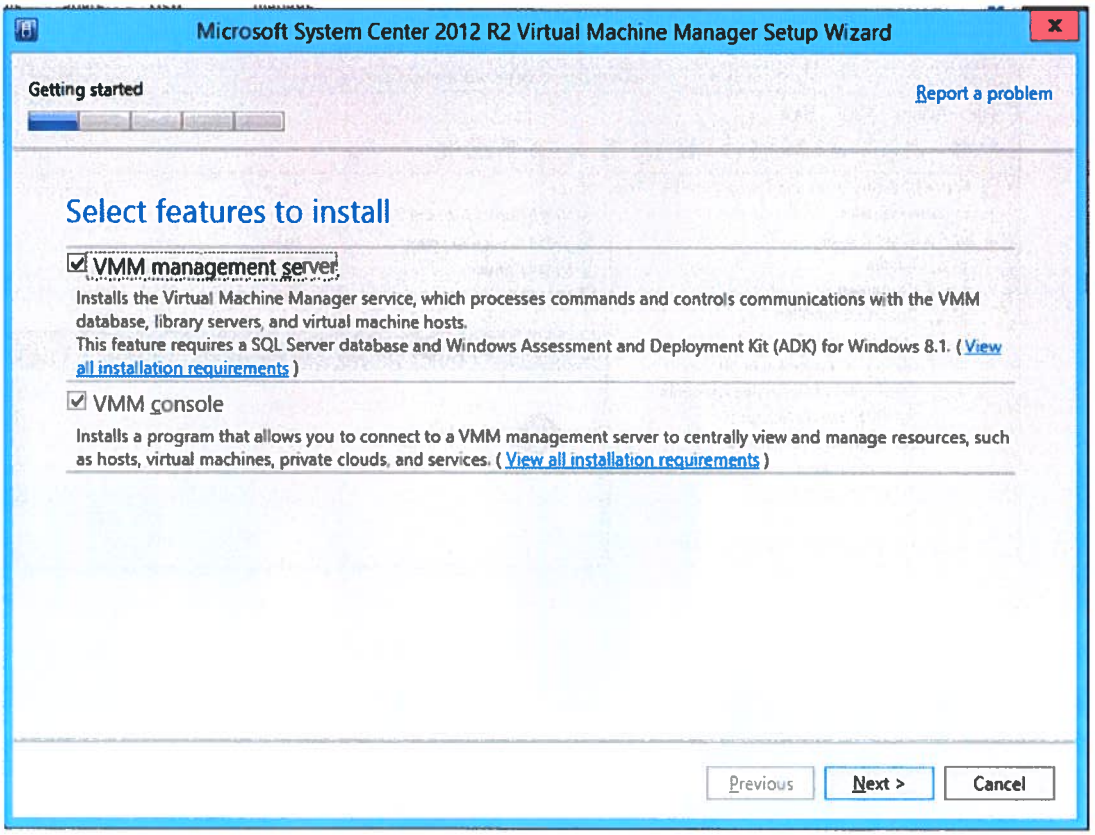


Figura 43 - Features para Instalar

Precisamos de registar o SCVMM.

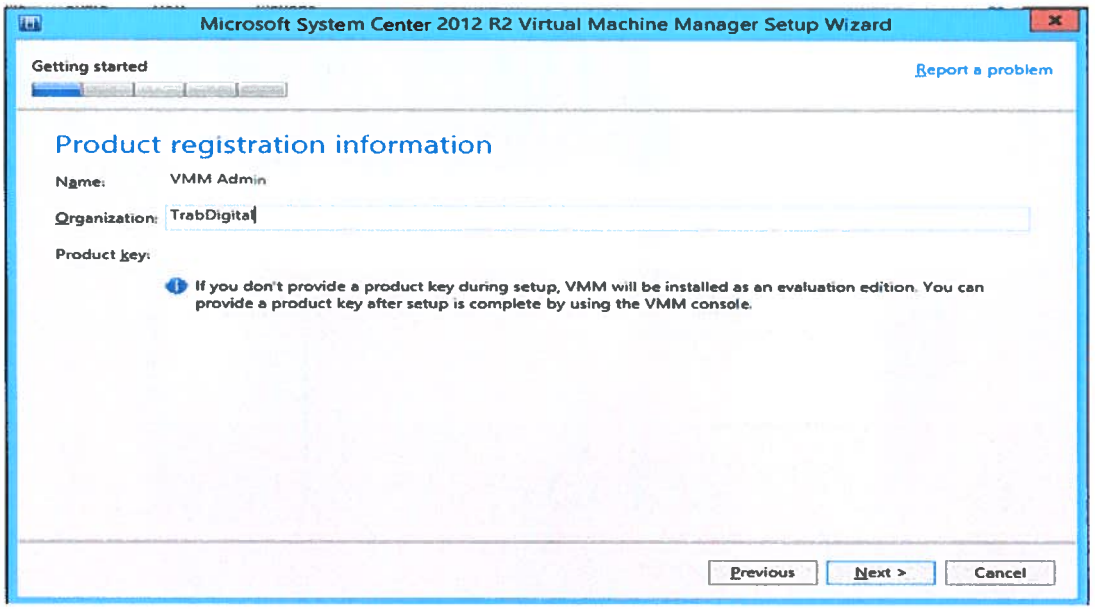


Figura 44 - Registo do SCVMM

Na máquina SQL abriram-se portas para haver comunicação a nível de protocolos transmission control protocol e user datagram protocol.

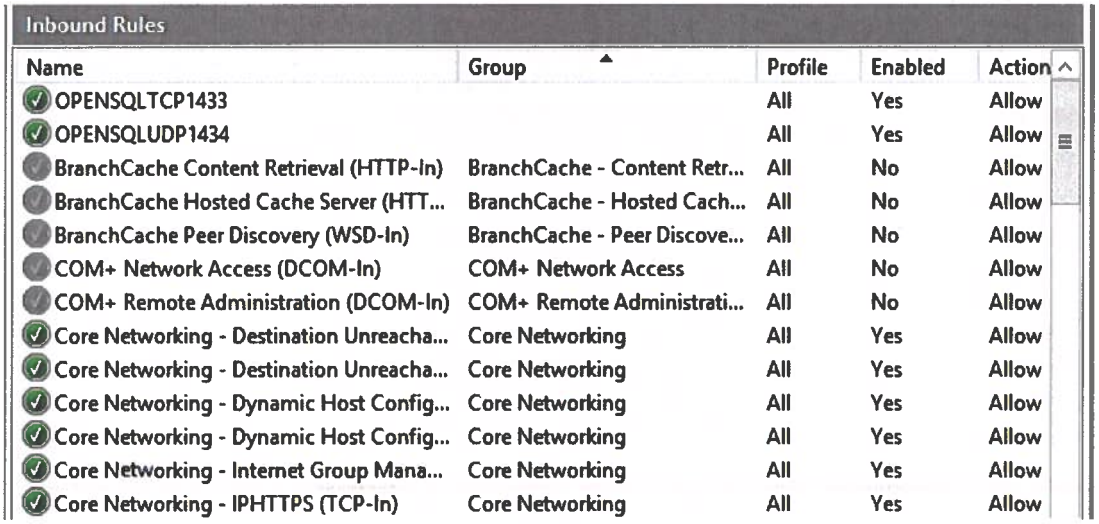


Figura 45 - Abrir Portas para estabelecer Ligação

Configuração do SCVMM para comunicar com a BaseDados e credenciais.

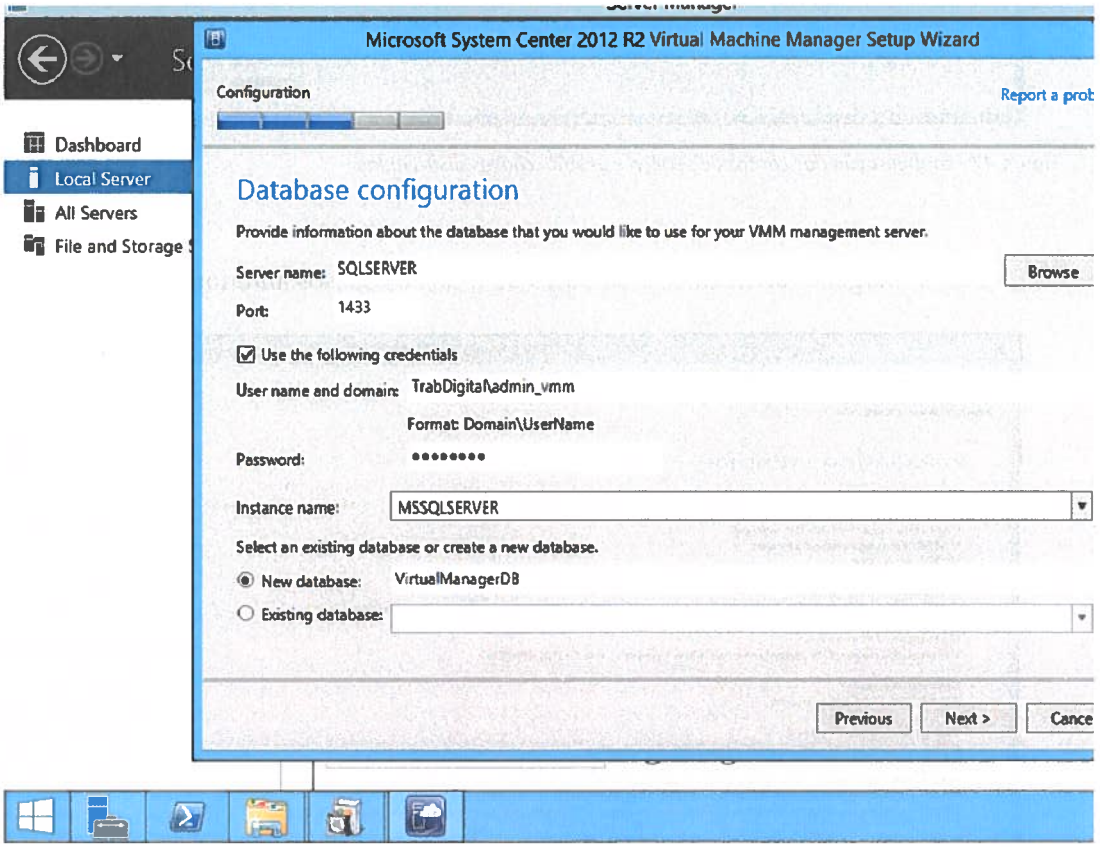


Figura 46 - Configuração SCVMM para Base Dados

Um dos passos mais importantes são as configurações das contas de serviço anteriormente referenciadas, e no controlador de domínio foi feito um contentor que é necessário para a gestão de chaves distribuídas.

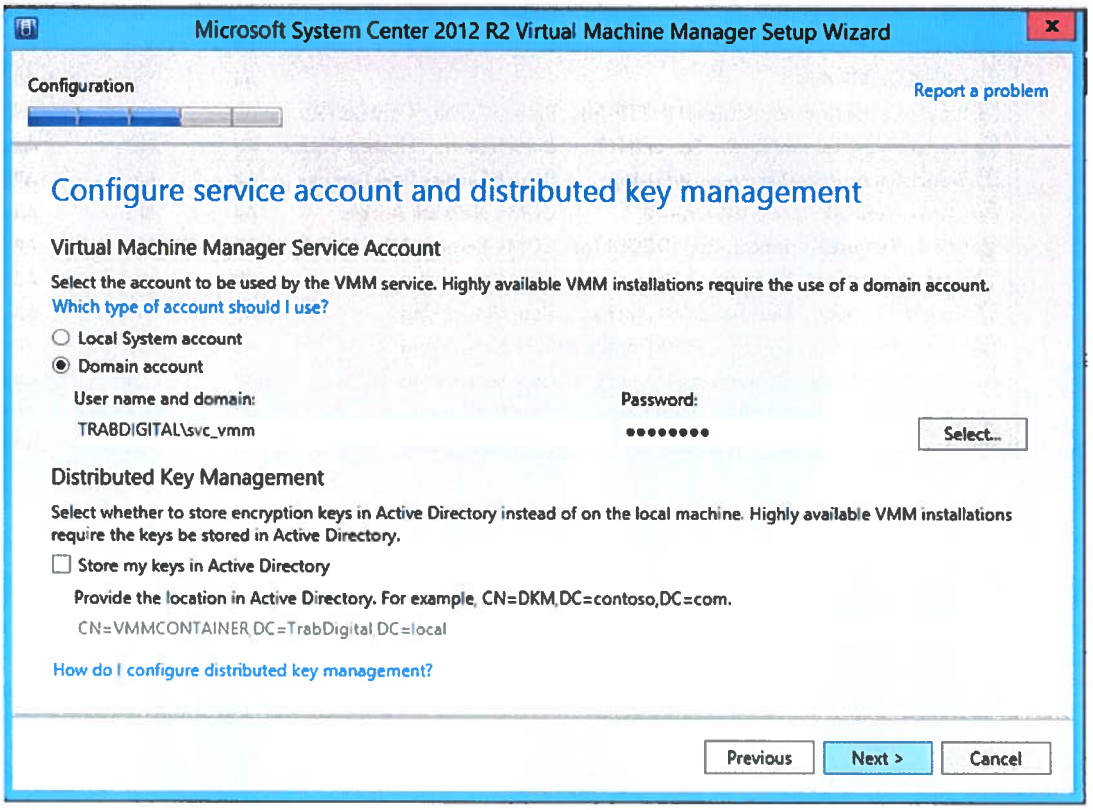


Figura 47 - configuração das contas de serviço e gestão chaves distribuidas

Nesta imagem verificamos o sumário de todos os passos anteriormente efetuados.

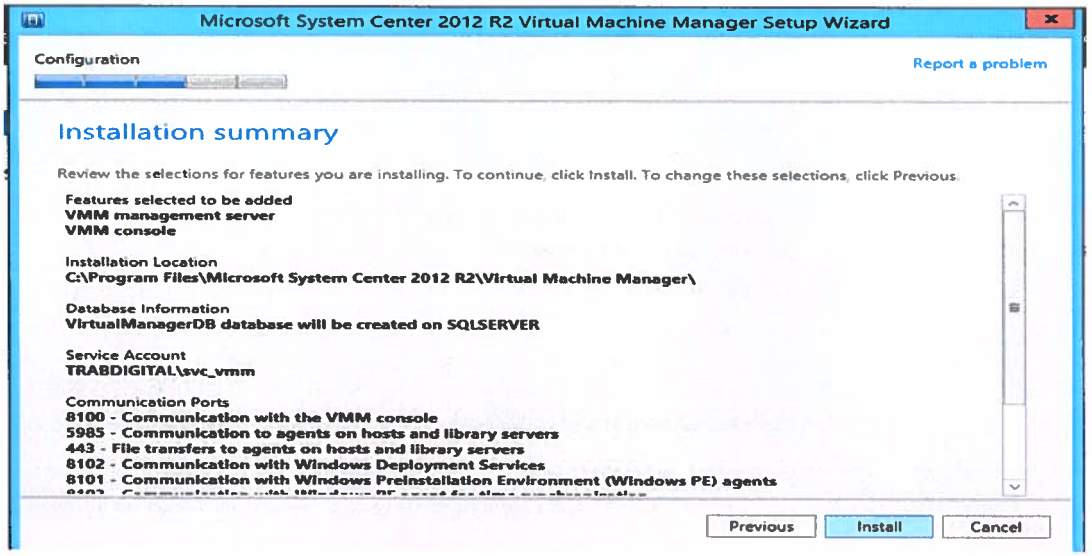


Figura 48 - Sumário

Para conseguirmos obter a máquina cliente dentro do Hyper-V, é necessário instalar a role na consola add roles and features wizard.

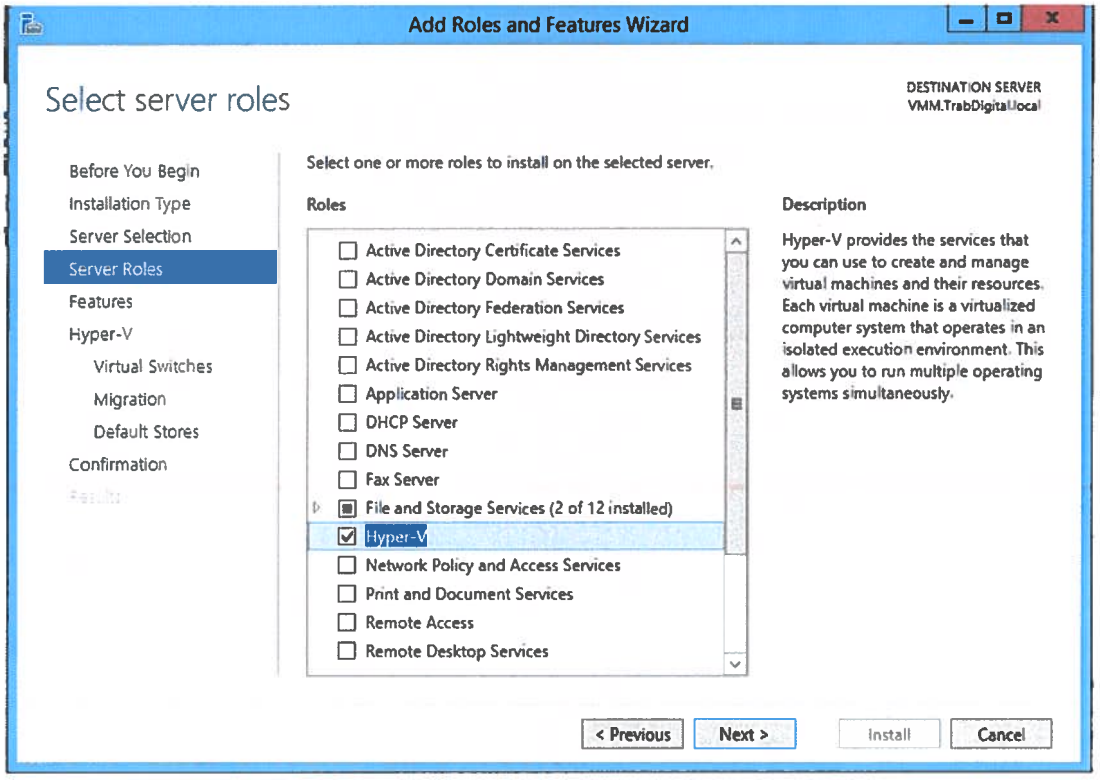


Figura 49 - Instalação Role Hyper-V

Na consola do SCVMM iremos proceder às especificações da máquina virtual cliente.

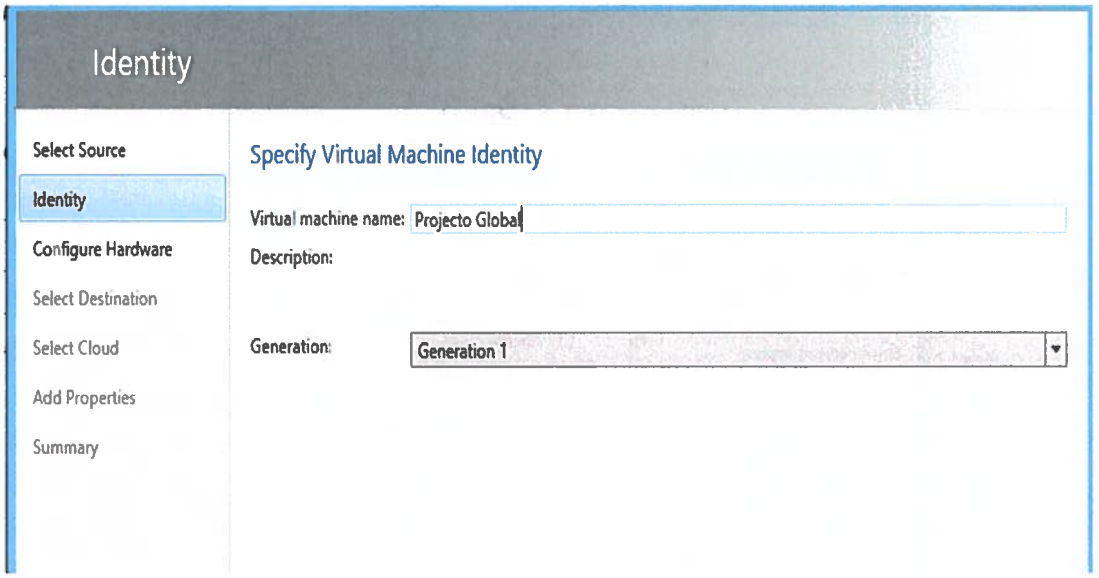


Figura 50 - Nome da Máquina Virtual Cliente

Numa primeira fase esta máquina vai servir como um template.

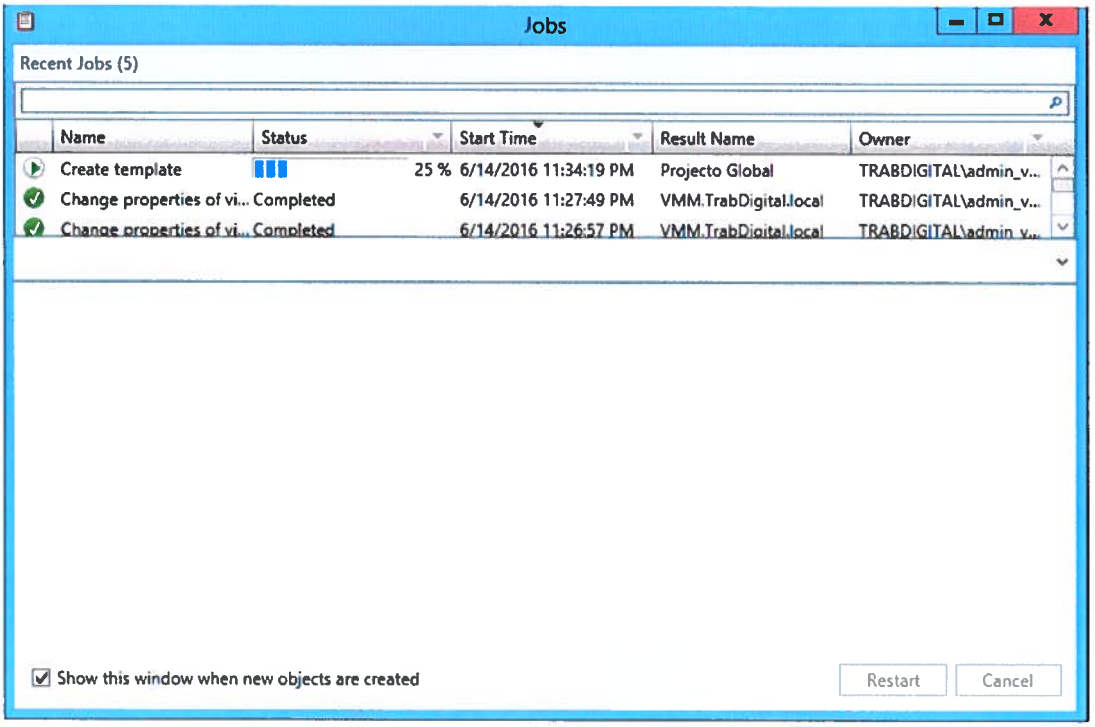


Figura 51 -Criação de um Template a partir de uma imagem iso do Windows 8.1

A partir da criação do template, criamos então a máquina virtual cliente, que irá servir de base para se instalar o Windows 8.1

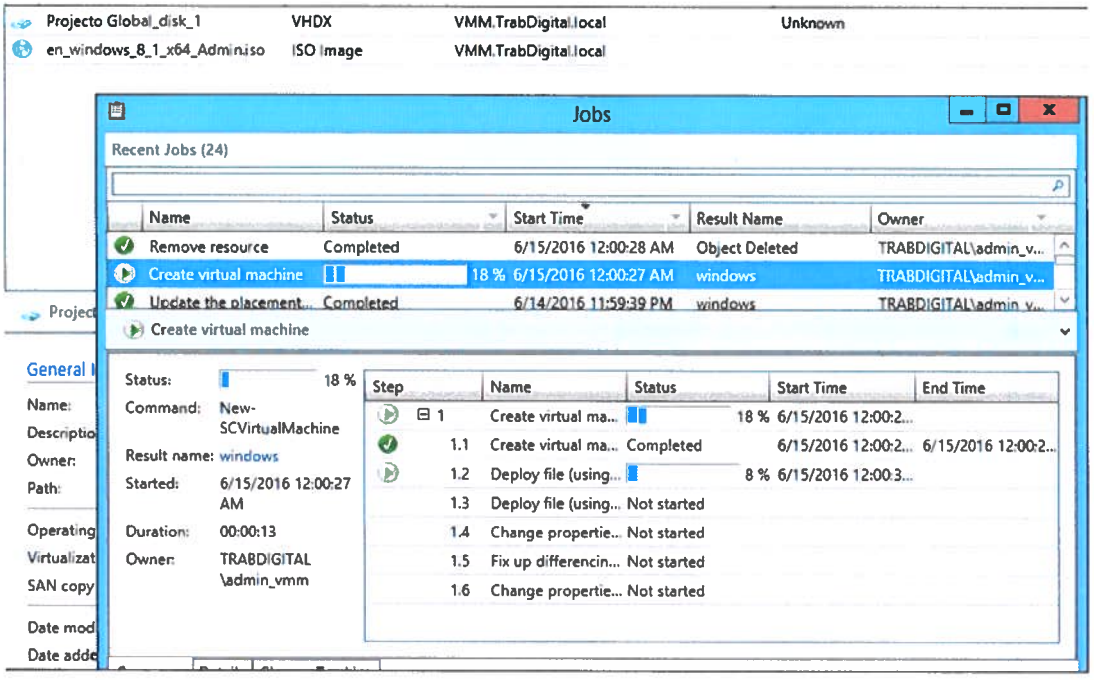


Figura 52 - Criação da Máquina Virtual baseada num Template

Acabando a sua criação, iremos associar a máquina denominada WINDOWS ao domínio, como se consta na imagem.

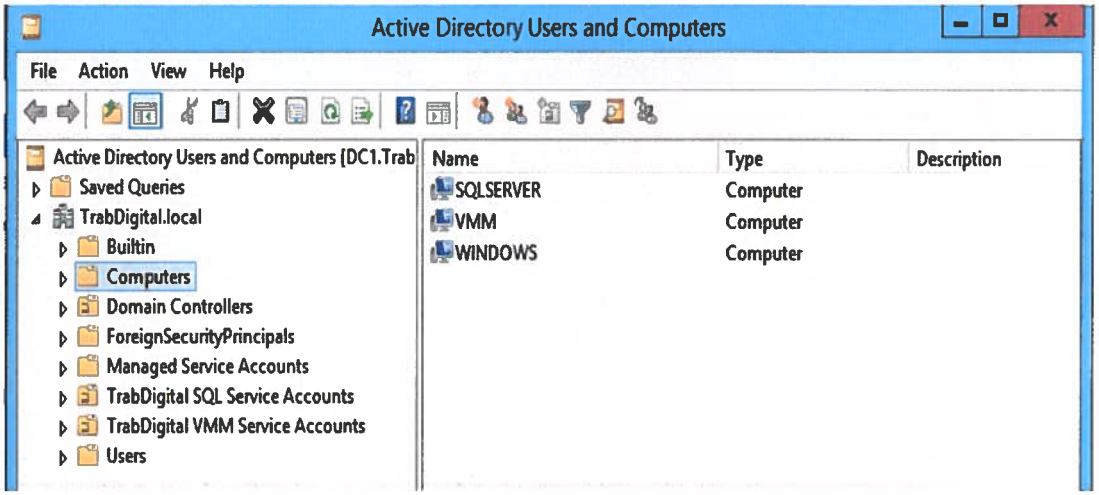


Figura 53 - Máquina Windows Associada ao Domínio

Máquina finalmente definida com todos os seus pré-requesitos de instalação (ip's definidos, nome atribuido, atualizações efetuadas), e fully qualified domain name.

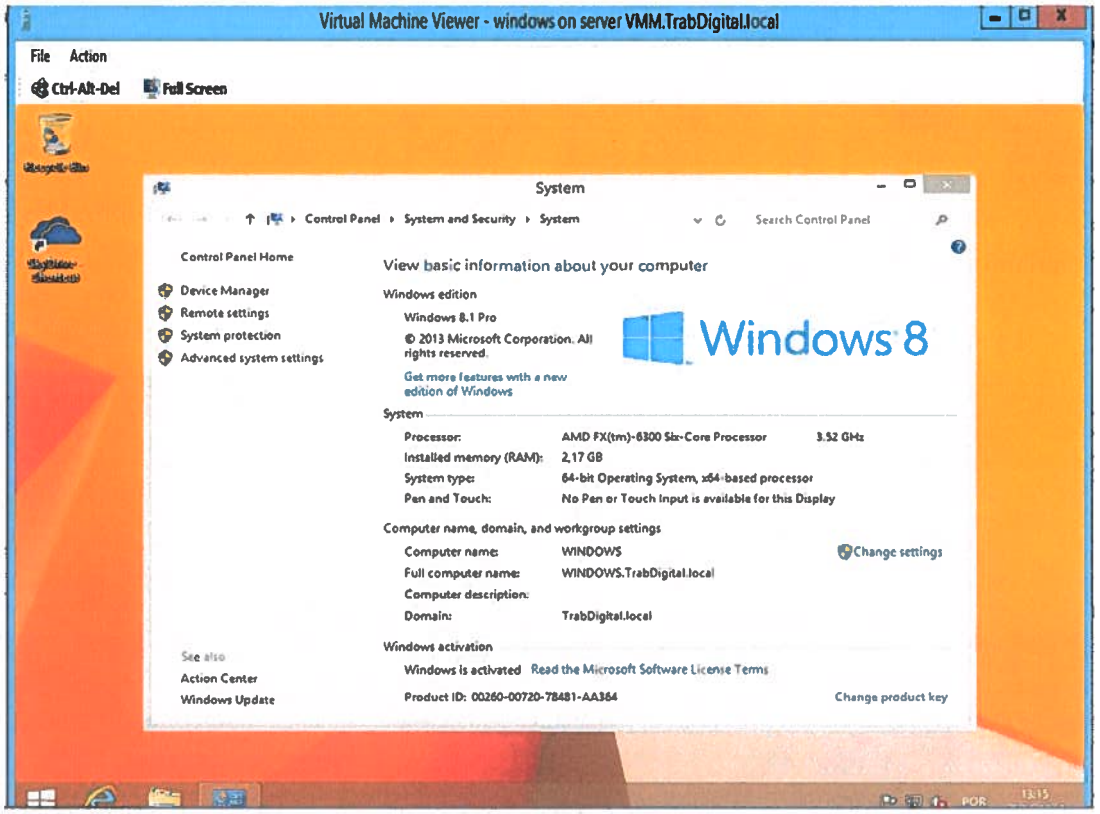


Figura 54 - Máquina Windows 8.1

Para este projeto, vamos efetuar a autenticação de duas maneiras, sendo uma delas com a conta já mencionada anteriormente, com privilégios de administrador do grupo VMMAdmins.

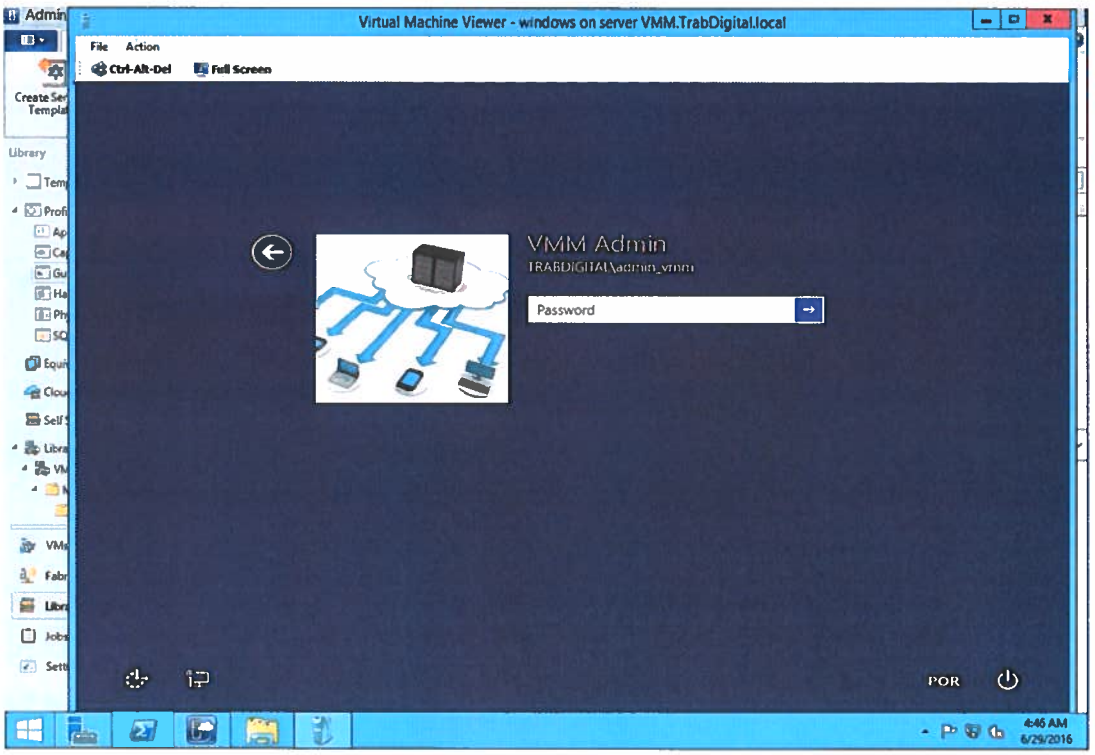


Figura 55 - Login como Administrador do grupo VMMAdmins

Para termos conhecimento da diferença dos privilégios de administrador no login, criamos tambem uma conta cliente.

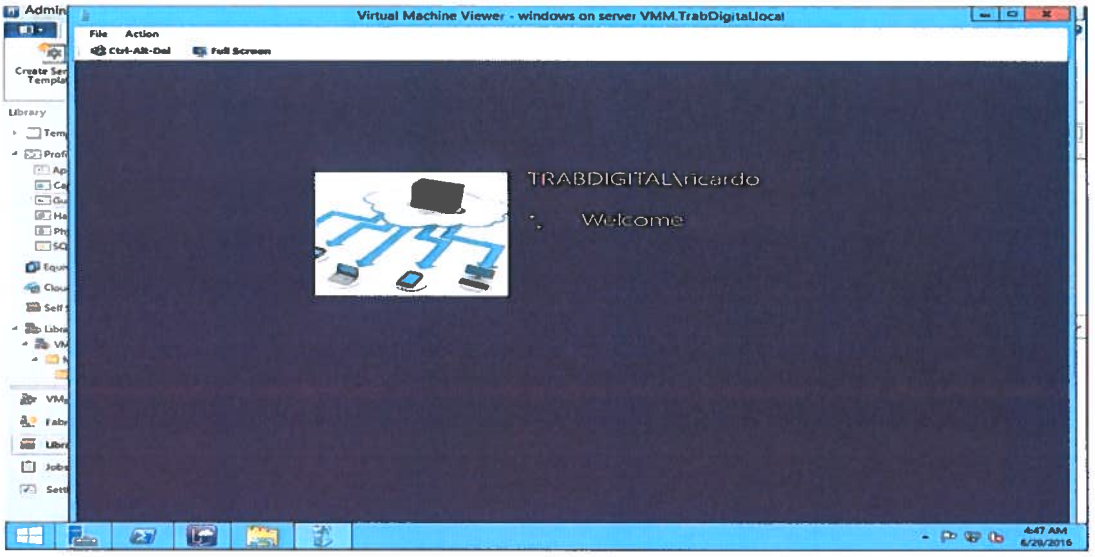


Figura 56 - Login sem poderes de administração

Por ultimo consegue-se aceder às máquinas remotamente do nosso controlador de domínio.

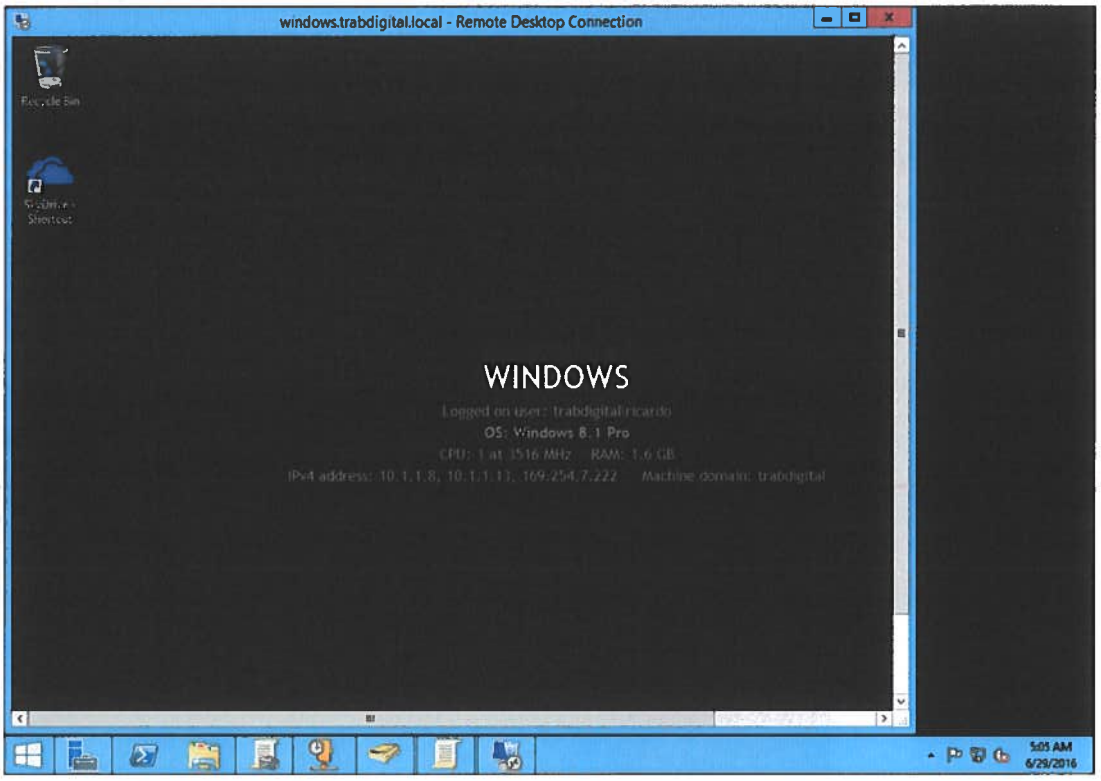


Figura 57 - Remote Desktop como Cliente

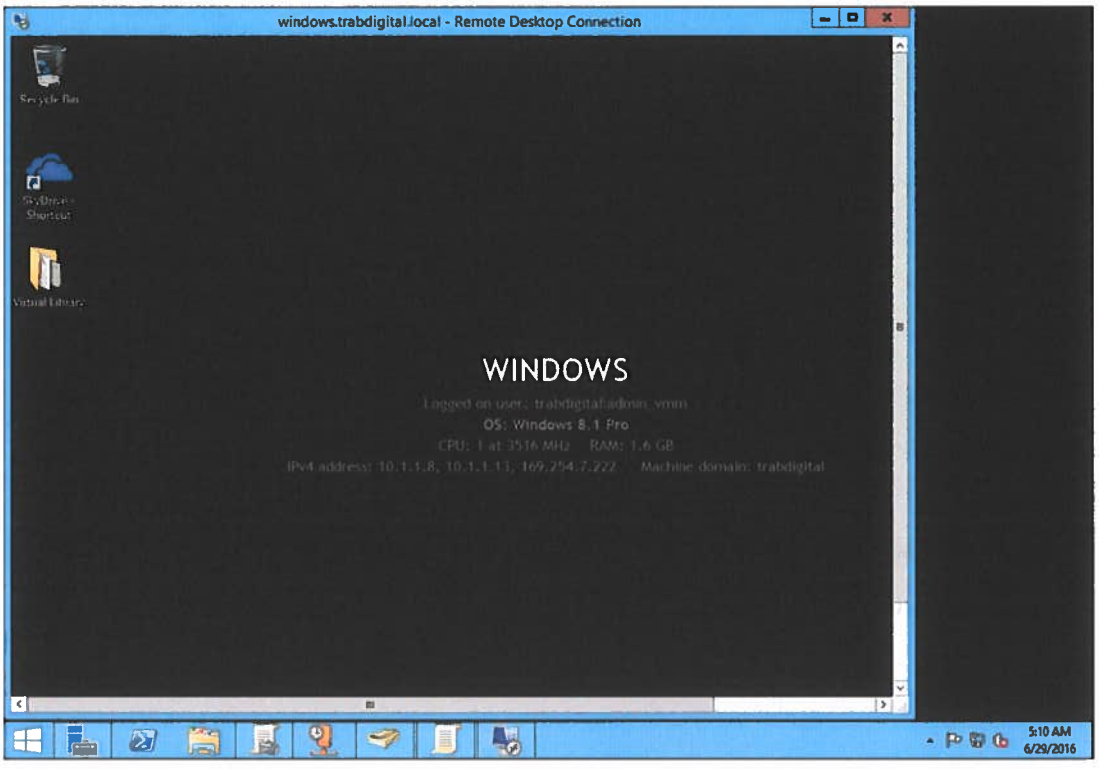


Figura 58 - Remote Desktop como Administrador

Aplicação TrabDigital

Neste quadro estão os utilizadores já logados, que fazem parte do dominio.

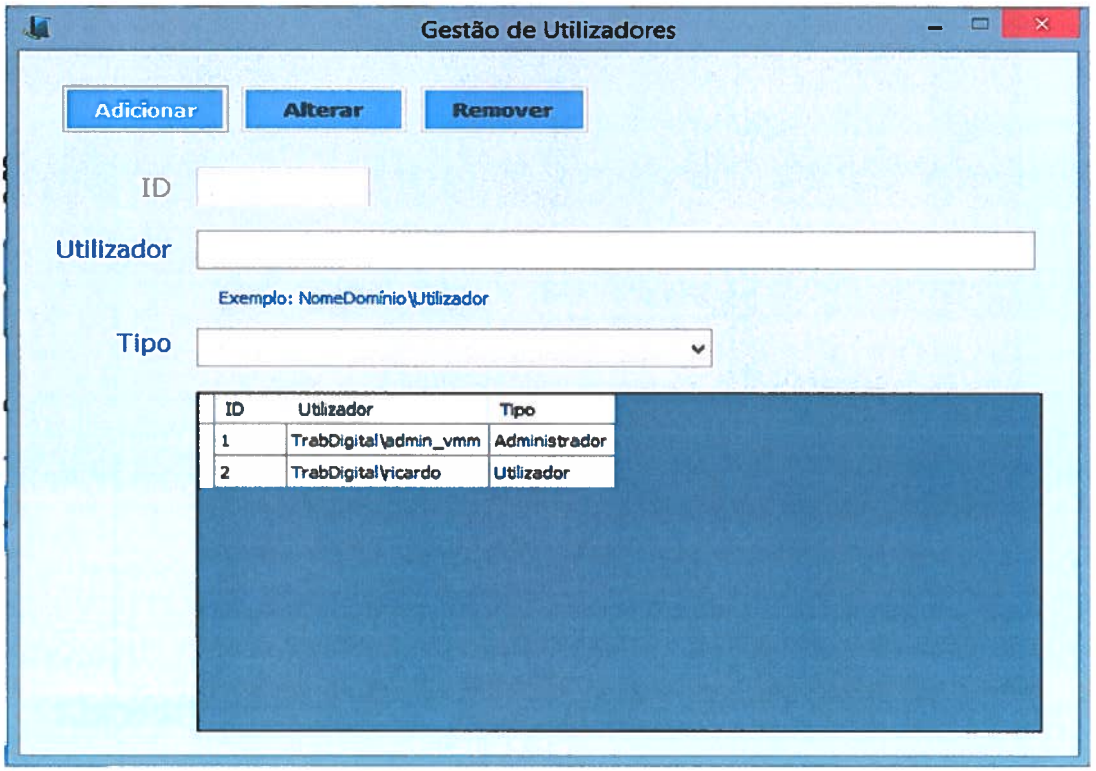


Figura 59 - Gestão de Utilizadores

No botão adicionar, o gestor da aplicação poderá introduzir o nome do novo utilizador que no seu entender possa fazer parte da sua biblioteca bem como o seu tipo, neste caso pode ter privilégios de administração ou não, o código poderá ser consultado no anexo IV ManageUsers.cs entre as linhas 166 a 205.

No botão alterar, o gestor poderá efetuar a sua alteração, o código poderá ser consultado no anexo IV ManageUsers.cs entre as linhas 210 a 243.

No botão remover, o gestor terá privilégios para remover os utilizadores que entenda o código poderá ser consultado no anexo IV ManageUsers.cs entre as linhas 246 e 288.

Na caixa de texto do utilizador podemos inserir os nomes desejados o código poderá ser consultado no anexo IV ManageUsers.cs entre as linhas 296 a 341.

Na caixa de texto do tipo o administrador poderá atribuir os privilégios que entender ao utilizador o código poderá ser consultado no anexo IV ManageUsers.cs entre as linhas 296 a 341.

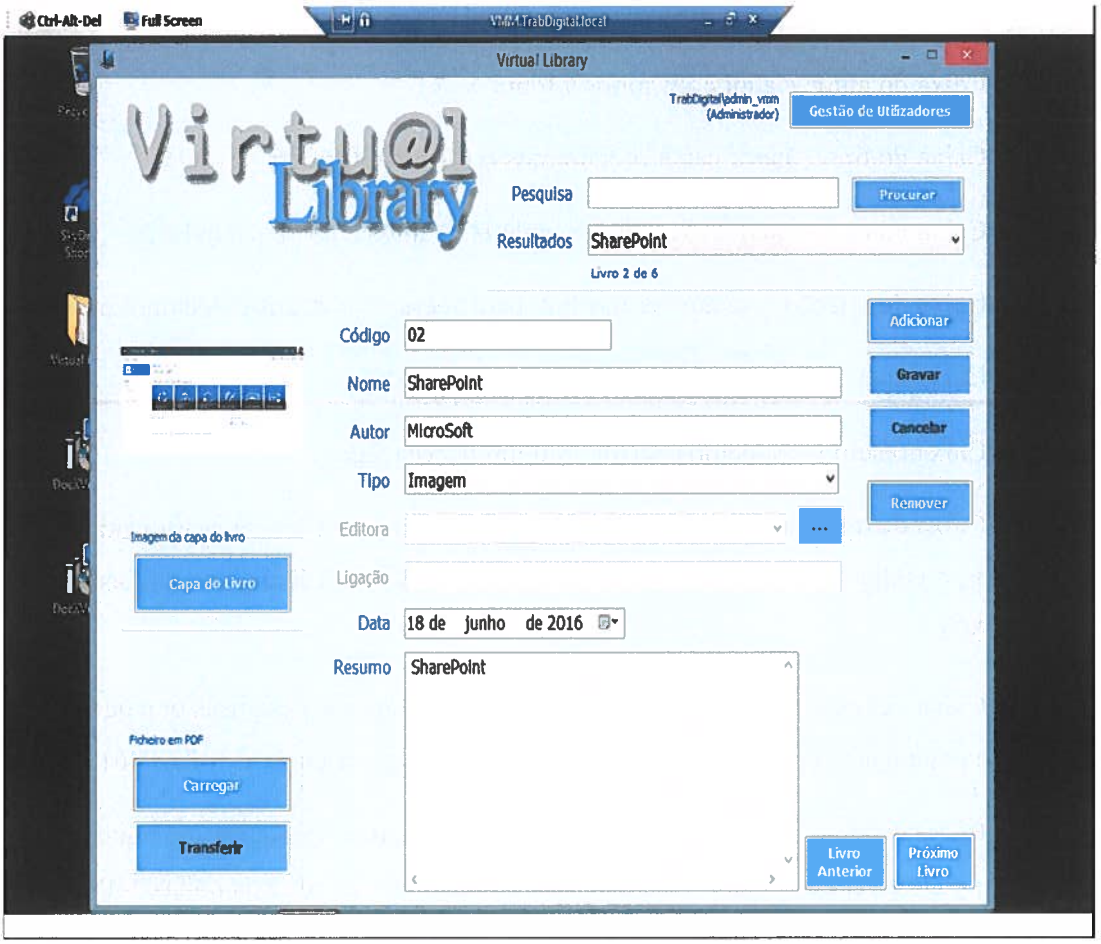


Figura 60 – TrabDigital (Virtual Library)

A imagem mostra-nos todo um processo, seja de pesquisa, de resultados da mesma, adicionar pdf's, imagens ou outros e, no seu entender pode-se efetuar á sua gravação, ou mesmo removê-la no caso de não a querer mais.

Podemos tambem carregar a capa do livro que pesquisamos ou transferir para uma pasta de destino.

Na imagem acima demonstrada, verifica-se que o administrador está logado na gestão de utilizadores, logo terá permissões para inserir dados, alterar os mesmos ou removê-los.

Caixa de texto pesquisa – insere-se o texto que se pretende pesquisar.

Caixa de texto resultados – verifica-se todo um historial de resultados gravados na base de dados.

Caixa de texto código – pode-se atribuir um código á pesquisa efetuada.

Caixa de texto nome – atribui-se um nome á pesquisa.

Caixa do autor – aqui colocamos a fonte.

Caixa do tipo – nesta caixa procuramos o formato da nossa pesquisa.

Caixa editora – qual será a editora onde efetuamos a nossa pesquisa.

Caixa de ligação – se houver um link para acesso aos dados colocamos aqui.

Caixa data – informa o utilizador do dia corrente.

Caixa resumo – podemos deixar aqui um breve resumo.

Caixa de texto gestão de utilizadores – nesta caixa verifica-se os utilizadores já logados, o código poderá ser consultado no anexo II VirtualLibrary.cs entre as linhas 819 e 828.

Caixa de texto procurar – nesta caixa fazemos todas as pesquisas que queremos, o código poderá ser consultado no anexo II VirtualLibrary.cs entre as linhas 461 a 478.

Caixa de texto adicionar – nesta caixa adicionamos as pesquisas que queremos, o código poderá ser consultado no anexo II VirtualLibrary.cs entre as linhas 480 a 492.

Caixa de texto gravar – nesta caixa o gestor grava ou não as pesquisas efetuadas, o código poderá ser consultado no anexo II VirtualLibrary.cs entre as linhas 510 e 750.

Caixa de texto cancelar – se no entender do utilizador num certo momento decidir cancelar a pesquisa, é aqui que poderá fazê-lo, o código poderá ser consultado no anexo II VirtualLibrary.cs entre as linhas 494 a 508.

Caixa de texto remover – caso o utilizador queira remover alguma pesquisa efetuada executa-la aqui nesta caixa, o código poderá ser consultado no anexo II VirtualLibrary.cs entre as linhas 924 e 976.

Caixa de texto da capa – para uma primeira abordagem o utilizador pode visualizar a capa, e se o texto for apelativo, pode consulta-lo, o código poderá ser consultado no anexo II VirtualLibrary.cs entre as linhas 777 e 815.

Caixa carregar os pdf´s – tendo na base de dados livros em pdf, aqui podemos carrega-los, o código poderá ser consultado no anexo II VirtualLibrary.cs entre as linhas 874 e 912.

Caixa de texto transferir – para se efetuar uma transferência é neste botão, o código poderá ser consultado no anexo II VirtualLibrary.cs entre as linhas 830 e 872.

8. Diagrama da Base de Dados

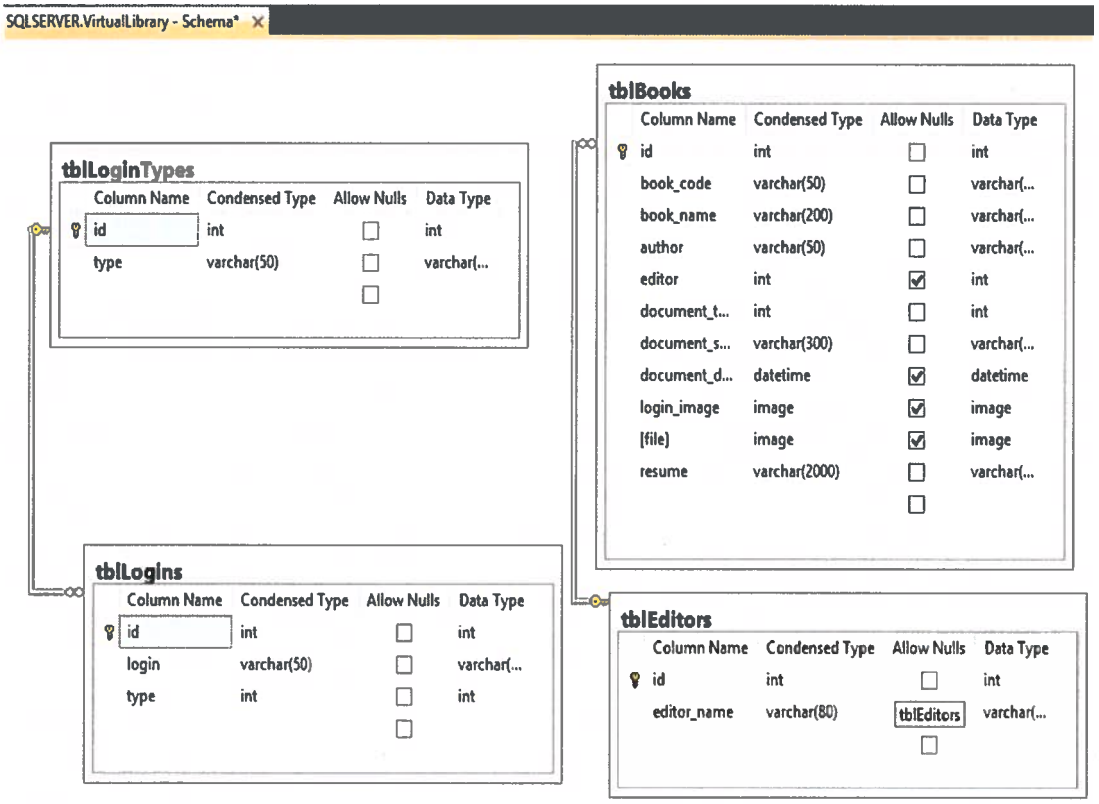


Figura 61 - Diagrama da Base de Dados

Tabela tbBooks

Campo id - Campo chave do tipo número inteiro com auto-incremento. Identificador de registo.

Campo book_code Campo do tipo varchar, suporta o código do livro.

Campo book_name Campo do tipo varchar, suporta o nome do livro.

Campo autor - Campo do tipo varchar até 50 caracteres.

Campo editor – Campo do tipo inteiro, suporta valores nulos.

Campo document_t – Campo de valores inteiros.

Campo document_s – Campo do tipo varchar, suporta até 300 caracteres.

Campo document_d – Campo do tipo data, suporta valores nulos.

Campo login_image – Campo do tipo imagem.

Campo file – Campo do tipo imagem.

Campo resume – Campo do tipo varchar, suporta até 200 caracteres.

TblEditors

Campo id - Campo chave do tipo número inteiro com auto-incremento. Identificador de registo.

Campo editor_name – Campo do tipo varchar.

TblLogins

Campo id - Campo chave do tipo número inteiro com auto-incremento. Identificador de registo.

Campo login - Campo do tipo varchar até 50 caracteres.

Campo tipo – Campo do tipo inteiro.

TbLoginTypes

Campo id - Campo chave do tipo número inteiro com auto-incremento. Identificador de registo.

Campo tipo – Campo do tipo varchar.

Bibliografia

Agarwal, Vidya Vrat (2012), Beginning C# 5.0 Databases.

Elmasri; Navathe, Fundamentals of Database Systems.

Ferreira, Nuno (2004), ISEP.

Gibson, Charles T., TIME-SHARING IN THE IBM SYSTEM/360:.

Goldworm, Barb, Server Virtualization, Ziff Davis paper

Hoopes, Jonh, Virtualization for Security : Including Sandboxing, Disaster Recovery, High Availability.

Hundhausen, Richard; Borg, Steven, (2002), Programming ADO.NET.

James R. Groff, Andrew J. Appel, SQL The Complete Reference third edition.

Jason Lefebre, Paul Bertucci, Sams teach yourself ADO.NET in 24 hours.

Kusnetzry, Dan, Virtualization A Manner`s Guide.

Lima, Júlio César Santos, (2012), O que é o SQL e qual a sua importância.

Marshall, David; Reynolds, Wade, Advanced Server Virtualization: VMware and Microsoft Platforms in the Virtual Data Center.

N.Weinberg, Paul, The Complete Reference SQL.

Technology, National Institute of Standards and

Wang, Wang, & Haung (2011).

Virtual Machine Feb. 76, international business machines corporation

Tulloch, Mitch, Understanding Microsoft Virtualization.

Troelsen, Andrew, Pro_CSharp_5.0_and_the_.NET_4.5_Framework_6th_edition.

Stephens, Ryan K.; Morgan, Bryan; Plew, Ronald, Teach Yourself SQL in 21 Days.

Santana, Gustavo, Data Center Virtualization.

Shah, Zahir Hussain, Windows server 2012.

O'Reilly, 2014.

Patrick, Tim, October 2010, Published by Microsoft Press. Microsoft ADO.NET 4.0 Step by Step.

Paul Nielsen, Uttam Parui, Microsoft SQL Server 2008 Bible.

Pearl, Robert, Healthy SQL: A Comprehensive Guide to Healthy SQL Server Performance.

Sousa, Correia &, (2010).

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

Conclusão

A aplicação TrabDigital, veio através deste trabalho desenvolvido demonstrar que as tecnologias de desenvolvimento que imergem no mercado podem minimizar e muito os custos elevados que as empresas têm para servir os seus clientes.

No caso da biblioteca digital, os recursos aplicados através da virtualização no projeto desenvolvido, traduz as suas enormes vantagens, como é comprovado na aplicaçãoaqui trabalhada. Foi desenvolvida por forma a estabelecer um nível seguro e estável, quer a nível de hardware, ou mesmo de ambiente gráfico, demonstrando os benifíciosque a virtualização pode trazer. O software utilizado, descrito neste projeto para suportar a aplicação, engloba as mais recentes tecnologias e ferramentas.

Foi assim possivel, por meio de uma consola virtual, aceder a uma plataforma, para armazenar e consultar na internet, por meio de links, livros em formato digital pdf e imagens. Efetua-se uma pesquisa mais rápida, consiza e sem perca de informação sendo que o fator de manutenção e desgaste dos livros devido ao seu manuseamento tenha sido eliminado.

ANEXOS

Anexo I – Functions.cs

```
1   using System;
2   using System.Collections.Generic;
3   using System.Configuration;
4   using System.Drawing;
5   using System.IO;
6   using System.Linq;
7   using System.Text;
8   using System.Windows.Forms;
9
10  namespace VirtualLibrary
11  {
12      public class functions
13      {
14          // Variáveis privadas a esta classe
15          private static int _UserID = 1;
16          private static string _Username;
17          private static int _UserType;
18
19          // Propriedades de sessão do utilizador
20          public static int UserID
21          {
22              get { return _UserID; }
23              set { _UserID = value; }
24          }
25          public static string Username
26          {
27              get { return _Username; }
28              set { _Username = value; }
29          }
30          public static int UserType
```

```
31          {
32              get { return _UserType; }
33              set { _UserType = value; }
34          }
35
36          // Variável para filtrar os ficheiros
37          public static string filtro = "Ficheiros PDF
(*.pdf)|*.pdf";
38
39          // Devolve a editora
40          internal static string GetDesc_Editor(int
editor_id)
41          {
42              //Liga à base de dados com um TableAdapter
43              DataSetsTableAdapters.tblEditorsTableAdapter
tutta = new DataSetsTableAdapters.tblEditorsTableAdapter();
44              tutta.Connection.ConnectionString =
Get_DBConnection();
45              //Carregar a tabela
46              DataSets.tblEditorsDataTable tutdt =
tutta.GetData_tblEditors(editor_id, null);
47
48              //Procurar a editora através do ID
49              DataSets.tblEditorsRow rowEditor =
tutdt.FindByid(editor_id);
50
51              //Se encontrou devolve a descrição associada
52              if (rowEditor != null)
53                  return rowEditor.editor_name;
54              else
55                  return null;
56          }
57
```

```
58          // Função para devolver o tipo de utilizador com
 base no ID fornecido
59          internal static string GetDesc_UserType(int
userType_id)
60          {
61              //Liga à base de dados com um TableAdapter
62              DataSetsTableAdapters.tblLoginTypesTableAdapter
tutta = new DataSetsTableAdapters.tblLoginTypesTableAdapter();
63              tutta.Connection.ConnectionString =
Get_DBConnection();
64              //Carregar a tabela
65              DataSets.tblLoginTypesDataTable tutdt =
tutta.GetData_tblLoginTypes(userType_id);
66
67              //Procurar o tipo de utilizador
68              DataSets.tblLoginTypesRow rowUT =
tutdt.FindByid(userType_id);
69
70              if (rowUT != null)
71                  return rowUT.type;
72              else
73                  return null;
74          }
75
76          // Correção de nome de ficheiro com carateres
inválidos
77          internal static string FixFilenameChars(string
filename)
78          {
79              string file_aux = filename;
80
81              //Remover carateres inválidos para ficheiro
82              file_aux = file_aux.Replace(@"\",
string.Empty);     //Remover '\'
```

```
83                file_aux = file_aux.Replace(@"/",
string.Empty);     //Remover '/'
84                file_aux = file_aux.Replace(@"*",
string.Empty);     //Remover '*'
85                file_aux = file_aux.Replace(@"?",
string.Empty);     //Remover '?'
86                file_aux = file_aux.Replace(@">",
string.Empty);     //Remover '>'
87                file_aux = file_aux.Replace(@"<",
string.Empty);     //Remover '<'
88                file_aux = file_aux.Replace("\"",
string.Empty);     //Remover '"'
89                file_aux = file_aux.Replace(@"|",
string.Empty);     //Remover '|'
90                file_aux = file_aux.Replace(@":",
string.Empty);     //Remover ':'
91                return file_aux;
92            }
93
94            // Selecionar um item
95            internal static void SelectComboItem(ComboBox cb,
int id)
96            {
97                for (int i = 0; i < cb.Items.Count; i++)
98                {
99                    KeyValuePair<string, int> valuePair =
((KeyValuePair<string, int>)cb.Items[i]);
100
101                   if (valuePair.Value == id)
102                   {
103                       cb.SelectedItem = valuePair;
104                       break;
105                   }
106               }
107           }
```

```
108
109         // Abrir ficheiro e retornar em byte[]
110         internal static byte[] OpenFile(string file_path,
string filters, string dialog_title)
111         {
112             byte[] picbyte = null;
113             bool askForImage = (file_path == null);
114
115             if (askForImage)
116             {
117                 //Procurar por uma imagem
118                 OpenFileDialog ofd = new OpenFileDialog();
119                 ofd.CheckFileExists = true;
120                 ofd.Filter = filters;
121                 ofd.Title = dialog_title;
122                 if (ofd.ShowDialog() == DialogResult.OK)
123                 {
124                     file_path = ofd.FileName;
125                 }
126             }
127
128             //Se o caminho existir
129             if (file_path != null && file_path !=
string.Empty)
130             {
131                 file_path =
System.IO.Path.GetFullPath(file_path);
132                 file_path = file_path.Replace(@"~\",
string.Empty);
133
134                 FileStream fs;
135                 fs = new FileStream(file_path,
FileMode.Open, FileAccess.Read);
136
```

```
137                     picbyte = new byte[fs.Length];
138
139                     fs.Read(picbyte, 0,
System.Convert.ToInt32(fs.Length));
140                     fs.Close();
141                 }
142                 return picbyte;
143             }
144
145             // obter imagem de um array de bytes
146             internal static Image GetImage_ByteArray(byte[]
bytes)
147             {
148                 MemoryStream ms = new MemoryStream(bytes);
149                 Image img = Image.FromStream(ms);
150                 return img;
151             }
152
153             // gravar ficheiro de array de bytes
154             internal static bool SaveByteArray_File(string
filename, byte[] bytes)
155             {
156                 try
157                 {
158                     //Abrir ficheiro para escrita
159                     System.IO.FileStream fs =
160                         new System.IO.FileStream(filename,
System.IO.FileMode.Create,
161
System.IO.FileAccess.Write);
162                     //Escreve um bloco de bytes usando os dados
do array de bytes
163                     fs.Write(bytes, 0, bytes.Length);
164
```

```
165                 //Fechar o ficheiro
166                 fs.Close();
167
168                 return true;
169             }
170             catch (Exception ex)
171             {
172                 //Erro
173                 Console.WriteLine("error: {0}",
174                                         ex.ToString());
175             }
176
177             //erro
178             return false;
179         }
180
181         // Obter a ligação à base de dadso
182         public static string Get_DBConnection()
183         {
184             string conn =
ConfigurationManager.ConnectionStrings["VirtualLibrary.Propertie
s.Settings.VirtualLibraryConnectionString"].ConnectionString;
185             return conn;
186         }
187     }
188 }
```

Anexo II – VirtualLibrary.cs

```
1   using System;
2   using System.Collections.Generic;
3   using System.ComponentModel;
4   using System.Configuration;
5   using System.Data;
6   using System.Drawing;
7   using System.Linq;
8   using System.Text;
9   using System.Threading.Tasks;
10  using System.Windows.Forms;
11
12  namespace VirtualLibrary
13  {
14      public partial class Biblioteca : Form
15      {
16          // Variáveis privadas
17          int current_book_id = -1;
18          int total_rows_found = 0;
19          private ControlsMode _mode;
20          bool loading_window = true;
21          bool hasBookPDF = false;
22
23          public enum ControlsMode
24          {
25              SearchBooks,    //Pesquisa
26              New_Book,       //Criação de um livro
27              Edit_Book,      //Edição do livro atual
28              Navigate        //Navegação
29          }
30          public ControlsMode mode
31          {
32              get
33              {
34                  return _mode;
35              }
36              set
37              {
38                  _mode = value;
39              }
40          }
41
42          // Construtor da janela
43          public Biblioteca()
44          {
```

```
45              //Inicialização dos controlos
46              InitializeComponent();
47
48              LoginWithCurrentUser();
49          }
50
51          // Iniciar sessão
52          private void LoginWithCurrentUser()
53          {
54              string loggedUser = Environment.UserDomainName
+ @"\" + Environment.UserName;
55
56              try
57              {
58                  //Liga à base de dados com um TableAdapter
59                  DataSetsTableAdapters.tblLoginsTableAdapter
tab_adap = new DataSetsTableAdapters.tblLoginsTableAdapter();
60                  tab_adap.Connection.ConnectionString =
functions.Get_DBConnection();
61                  //Carregar a tabela
62                  DataSets.tblLoginsDataTable dtab =
tab_adap.GetData_tblLogins(null, loggedUser, null);
63
64                  //Se existe o utilizador e a password é a
correta
65                  if (dtab.Rows.Count == 1)
66                  {
67                      DataSets.tblLoginsRow userRow =
((DataSets.tblLoginsRow)dtab.Rows[0]);
68
69                      //Liga à base de dados com um
TableAdapter
70
DataSetsTableAdapters.tblLoginsTableAdapter tab_adap2 = new
DataSetsTableAdapters.tblLoginsTableAdapter();
71                      tab_adap2.Connection.ConnectionString =
functions.Get_DBConnection();
72                      //Carregar a tabela
73                      DataSets.tblLoginsDataTable dtab2 =
tab_adap2.GetData_tblLogins(userRow.id, null, null);
74
75                      //Atribuir as informações do utilizador
às variáveis globais
76                      functions.UserID = userRow.id;
77                      functions.Username = userRow.login;
```

```
78                      functions.UserType = userRow.type;
79
80                      l_UserLogged.Text = functions.Username
+ "\r\n";
81                      l_UserLogged.Text +=
(functions.UserType == 1 ? "(Administrador)" : "(Utilizador)");
82                  }
83                  else
84                  {
85                      //Erro a iniciar sessão / Utilizador ou
password errados
86                      DialogResult result =
MessageBox.Show("Erro ao iniciar sessão\r\n\r\nO utilizador " +
functions.Username + " não tem permissão para utilizar esta
aplicação", "Sem acesso", MessageBoxButtons.OK,
MessageBoxIcon.Exclamation);
87                      Environment.Exit(0);
88                  }
89              }
90              catch (Exception ex)
91              {
92                  MessageBox.Show("Erro no início de
sessão.\r\nErro: " + ex.Message, "Erro ao iniciar sessão",
MessageBoxButtons.OK, MessageBoxIcon.Exclamation);
93              }
94          }
95
96          // Carregamento da janela
97          private void Biblioteca_Load(object sender,
EventArgs e)
98          {
99              //Imagens 'Default' da pré-visualização do
livro e da photo de utilizador
100             //          imgBookCover.Image = new
Image(new Uri("/Images/BookNotAvailable.png",
UriKind.Relative));
101             //          imgUserPhoto.Source = new
Image(new Uri("/Images/no_person_available.png",
UriKind.Relative));
102
103             //Carregamento das editoras
104             LoadEditors();
105
106             //Carregamento dos tipos de documentos
107             LoadDocTypes();
```

```
108
109             c_DocType.ctl_CBox.SelectedIndexChanged +=
c_DocType_ctl_CBox_SelectedIndexChanged;
110             t_source.ctl_TBox.TextChanged +=
ctl_TBox_TextChanged;
111             d_datetime.ctl_DTime.TextChanged +=
ctl_DTime_TextChanged;
112
113             //Variável para detetar que o carregamento
terminou
114             loading_window = false;
115
116             //Iniciar como modo de pesquisa
117             mode = ControlsMode.SearchBooks;
118
119             c_Books.ctl_CBox.SelectedIndexChanged +=
c_Books_SelectionChanged;
120
121             t_Reference.ctl_TBox.TextChanged +=
t_Reference_TextChanged;
122             t_Title.ctl_TBox.TextChanged +=
t_Title_TextChanged;
123             t_Author.ctl_TBox.TextChanged +=
t_Author_TextChanged;
124             t_BookDescription.TextChanged +=
t_BookDescription_TextChanged;
125             c_Editor.ctl_CBox.SelectedIndexChanged +=
c_Editor_SelectedIndexChanged;
126
127             //Disponibilizar campos
128             SetControls();
129         }
130
131         private void ctl_DTime_TextChanged(object sender,
EventArgs e)
132         {
133             //Se o modo é diferente de novo livro
134             if (mode != ControlsMode.New_Book)
135                 mode = ControlsMode.Edit_Book;   //Colocar
o modo como edição
136             SetControls();
137         }
138
139         private void ctl_TBox_TextChanged(object sender,
EventArgs e)
```

```
140         {
141             //Se o modo é diferente de novo livro
142             if (mode != ControlsMode.New_Book)
143                 mode = ControlsMode.Edit_Book;   //Colocar
o modo como edição
144             SetControls();
145         }
146
147         // Carregamento das editoras na combo
148         private void LoadEditors()
149         {
150             //Liga à base de dados com um TableAdapter
151             DataSetsTableAdapters.tblEditorsTableAdapter
tpta = new DataSetsTableAdapters.tblEditorsTableAdapter();
152             tpta.Connection.ConnectionString =
functions.Get_DBConnection();
153             //Carregar a tabela
154             DataSets.tblEditorsDataTable tpdt =
tpta.GetData_tblEditors(null, null);
155
156             //Limpar a datagrid
157             c_Editor.ctl_CBox.Items.Clear();
158
159             //Carregamento da combo
160             c_Editor.ctl_CBox.DisplayMember = "Key";
161             c_Editor.ctl_CBox.ValueMember = "Value";
162             c_Editor.ctl_CBox.Items.Add(new
KeyValuePair<string, int>("", -1));
163             foreach (DataSets.tblEditorsRow row in
tpdt.Rows)
164                 c_Editor.ctl_CBox.Items.Add(new
KeyValuePair<string, int>(row.editor_name, row.id));
165
166             //Se houver pelo menos 1 registo, selecioná-lo
167             if (c_Editor.ctl_CBox.Items.Count > 0)
c_Editor.ctl_CBox.SelectedIndex = 0;
168         }
169         // Carregamento do livro selecionado
170         private void LoadCurrentSelectedBook()
171         {
172             //Se não houver um livro selecionado
173             if (c_Books.ctl_CBox.SelectedItem == null)
174             {
175                 //Limpar os controlos
176                 t_Reference.ctl_TBox.Text = string.Empty;
```

```
177                     t_Title.ctl_TBox.Text = string.Empty;
178                     t_Author.ctl_TBox.Text = string.Empty;
179                     c_Editor.ctl_CBox.SelectedIndex = -1;
180                     c_DocType.ctl_CBox.SelectedIndex = 0;
181                     t_source.ctl_TBox.Text = string.Empty;
182                     d_datetime.ctl_DTime.Value = DateTime.Now;
183                     t_BookDescription.Text = string.Empty;
184 
185                     //Carregar imagem de livro não disponível
186                     byte[] picbyte_empty =
functions.OpenFile(@"~/Images/BookNotAvailable.png", "Image
Files (*.bmp; *.jpg; *.png; *.gif)|*.bmp; *.jpg; *.png; *.gif",
"Escolher uma imagem para o livro");
187                     i_BookCover.Image =
functions.GetImage_ByteArray(picbyte_empty);
188 
189                     //Não existe PDF do livro
190                     hasBookPDF = false;
191 
192                     //Sair da função
193                     return;
194                 }
195 
196                 //Obter o ID do livro selecionado
197                 int book_id = ((KeyValuePair<string,
int>)c_Books.ctl_CBox.SelectedItem).Value;
198 
199                 //Liga à base de dados com um TableAdapter
200                 DataSetsTableAdapters.tblBooksTableAdapter tbta
= new DataSetsTableAdapters.tblBooksTableAdapter();
201                 tbta.Connection.ConnectionString =
functions.Get_DBConnection();
202                 //Carregar a tabela
203                 DataSets.tblBooksDataTable tbdt =
tbta.GetData_tblBooks(book_id, null, null, null, null, null,
null, null, null);
204 
205                 //Se encontrou o registo
206                 if (tbdt != null && tbdt.Rows.Count > 0)
207                 {
208                     DataSets.tblBooksRow row =
((DataSets.tblBooksRow)tbdt.Rows[0]);
209                     if (row != null)
210                     {
```

```
211                         //Guardar o modo para mais tarde o
voltar a obter
212                         ControlsMode old_mode = mode;
213
214                         //Colocar os valores nos respetivos
controlos
215                         t_Reference.ctl_TBox.Text =
row.book_code;
216                         t_Title.ctl_TBox.Text = row.book_name;
217                         t_Author.ctl_TBox.Text = row.author;
218
functions.SelectComboItem(c_DocType.ctl_CBox,
row.document_type);
219                         if (row.IseditorNull())
220
functions.SelectComboItem(c_Editor.ctl_CBox, -1);
221                         else
222
functions.SelectComboItem(c_Editor.ctl_CBox, row.editor);
223
c_Editor_SelectedIndexChanged(c_Editor.ctl_CBox, null);
224                         t_source.ctl_TBox.Text =
row.document_source;
225                         d_datetime.ctl_DTime.Value =
row.document_date;
226                         t_BookDescription.Text = string.Empty;
227                         t_BookDescription.Text = row.resume;
228
229                         //Variável que mostra se tem ou não um
PDF
230                         hasBookPDF = (row.IsfileNull() ==
false);
231
232                         //Se existe uma pré-visualização do
livro
233                         if (row.Islogin_imageNull() == false)
234                         {
235                             //Abrir a imagem
236                             byte[] picbyte = row.login_image;
237                             i_BookCover.Image =
functions.GetImage_ByteArray(row.login_image);
238                         }
239                         else
240                         {
```

```
241                             //Abrir uma imagem de capa não
disponível
242                             byte[] picbyte_empty =
functions.OpenFile(@"~/Images/BookNotAvailable.png", "Image
Files (*.bmp; *.jpg; *.png; *.gif)|*.bmp; *.jpg; *.png; *.gif",
"Escolher uma imagem para o livro");
243                             i_BookCover.Image =
functions.GetImage_ByteArray(picbyte_empty);
244                         }
245
246                         //Recolocar o modo guardado
247                         mode = old_mode;
248                     }
249                 }
250             }
251             // Carregamento de Tipos de Documentos
252             private void LoadDocTypes()
253             {
254                 //Limpar a datagrid
255                 c_DocType.ctl_CBox.Items.Clear();
256
257                 //Carregamento da combo
258                 c_DocType.ctl_CBox.DisplayMember = "Key";
259                 c_DocType.ctl_CBox.ValueMember = "Value";
260
261                 //Carregamento da combo
262                 c_DocType.ctl_CBox.Items.Add(new
KeyValuePair<string, int>("Livro", 1));
263                 c_DocType.ctl_CBox.Items.Add(new
KeyValuePair<string, int>("Jornal", 2));
264                 c_DocType.ctl_CBox.Items.Add(new
KeyValuePair<string, int>("Revista", 3));
265                 c_DocType.ctl_CBox.Items.Add(new
KeyValuePair<string, int>("Imagem", 4));
266                 c_DocType.ctl_CBox.Items.Add(new
KeyValuePair<string, int>("Manuscrito", 5));
267                 c_DocType.ctl_CBox.Items.Add(new
KeyValuePair<string, int>("Link da Internet", 6));
268
269                 c_DocType.ctl_CBox.SelectedIndex = 0;
270             }
271
272             // Preencher combo com os livros
273             private void FillComboOfBooksFound()
274             {
```

```
275                 //Guardar o modo para mais tarde o voltar a
obter
276                 ControlsMode old_mode = mode;
277
278                 //Liga à base de dados com um TableAdapter
279
DataSetsTableAdapters.tblBooksByFindTableAdapter bfbata = new
DataSetsTableAdapters.tblBooksByFindTableAdapter();
280                 bfbata.Connection.ConnectionString =
functions.Get_DBConnection();
281                 //Carregar a tabela
282                 DataSets.tblBooksByFindDataTable bfbadt =
bfbata.GetData_tblBooksByFind("%" + t_Search.ctl_TBox.Text +
"%");
283
284                 //Limpar os registos do controlo
285                 c_Books.ctl_CBox.Items.Clear();
286
287                 //Se não encontrou registos
288                 if (bfbadt.Rows.Count == 0)
289                 {
290                     MessageBox.Show("Não foram encontrados
registos para o critério de pesquisa introduzido", "Pesquisa sem
registos", MessageBoxButtons.OK, MessageBoxIcon.Information);
291                 }
292                 else  //Se não... carregar os registos
encontrados
293                 {
294                     //Carregamento do controlo
295                     c_Books.ctl_CBox.DisplayMember = "Key";
296                     c_Books.ctl_CBox.ValueMember = "Value";
297                     foreach (DataSets.tblBooksByFindRow row in
bfbadt.Rows)
298                         c_Books.ctl_CBox.Items.Add(new
KeyValuePair<string, int>(row.book_name, row.id));
299                 }
300
301                 //Contagem do total de livros encontrados
302                 total_rows_found = bfbadt.Rows.Count;
303                 //Índice atual caso tenha ou não encontrado
registos
304                 current_book_id = (bfbadt.Rows.Count >= 1 ? 0 :
-1);
305
306                 //Selecionar o 1º livro encontrado
```

```
307                 c_Books.ctl_CBox.SelectedIndex =
current_book_id;
308
309                 //Recolocar o modo guardado
310                 mode = old_mode;
311
312                 //Disponibilizar campos
313                 SetControls();
314             }
315
316             // Disponibilizar campos
317             private void SetControls()
318             {
319                 if (loading_window) return;
320
321                 t_Search.Enabled = (mode ==
ControlsMode.SearchBooks || mode == ControlsMode.Navigate);
322                 b_Search.Enabled = (mode ==
ControlsMode.SearchBooks || mode == ControlsMode.Navigate);
323
324                 c_Books.Enabled = (total_rows_found > 0) &&
(mode == ControlsMode.SearchBooks || mode ==
ControlsMode.Navigate);
325                 c_Editor.Enabled = (total_rows_found > 0) &&
(mode == ControlsMode.SearchBooks || mode ==
ControlsMode.Navigate);
326                 b_Cancel.Enabled = ((mode ==
ControlsMode.Edit_Book || mode == ControlsMode.New_Book) ? true
: false);
327
328                 //Se o utilizador é administrador
329                 if (functions.UserType == 1)
330                 {
331                     if (total_rows_found == 0)
332                     {
333                         //Se não foram encontrados registos
334                         t_Reference.Enabled = (mode ==
ControlsMode.New_Book || mode == ControlsMode.Edit_Book);
335                         t_Title.Enabled = (mode ==
ControlsMode.New_Book || mode == ControlsMode.Edit_Book);
336                         t_Author.Enabled = (mode ==
ControlsMode.New_Book || mode == ControlsMode.Edit_Book);
337                         c_Editor.Enabled = (mode ==
ControlsMode.New_Book || mode == ControlsMode.Edit_Book);
```

```
338                         c_DocType.Enabled = (mode ==
ControlsMode.New_Book || mode == ControlsMode.Edit_Book);
339                         t_source.Enabled = (mode ==
ControlsMode.New_Book || mode == ControlsMode.Edit_Book);
340
341                         int doc_type_selected =
((KeyValuePair<string,
int>)c_DocType.ctl_CBox.SelectedItem).Value;
342                         //Disponibilizar se for um livro
343                         c_Editor.Enabled = (doc_type_selected
== 1) && mode != ControlsMode.SearchBooks;
344                         t_source.Enabled = (doc_type_selected
== 6) && mode != ControlsMode.SearchBooks;
345
346                         d_datetime.Enabled = (mode ==
ControlsMode.New_Book || mode == ControlsMode.Edit_Book);
347                         l_ResumeOfBook.Enabled = (mode ==
ControlsMode.New_Book || mode == ControlsMode.Edit_Book);
348                         t_BookDescription.Enabled = (mode ==
ControlsMode.New_Book || mode == ControlsMode.Edit_Book);
349                     }
350                     else
351                     {
352                         //Se foram encontrados registos
353                         t_Reference.Enabled = true;
354                         t_Title.Enabled = true;
355                         t_Author.Enabled = true;
356                         c_Editor.Enabled = true;
357                         c_DocType.Enabled = true;
358                         t_source.Enabled = true;
359
360                         int doc_type_selected =
((KeyValuePair<string,
int>)c_DocType.ctl_CBox.SelectedItem).Value;
361                         c_Editor.Enabled = (doc_type_selected
== 1) && mode != ControlsMode.SearchBooks;
362                         t_source.Enabled = (doc_type_selected
== 6) && mode != ControlsMode.SearchBooks;
363
364                         d_datetime.Enabled = true;
365                         l_ResumeOfBook.Enabled = true;
366                         t_BookDescription.Enabled = true;
367                     }
368                 }
369                 else //Se o utilizador não é administrador
```

```
370             {
371                 t_Reference.Enabled = false;
372                 t_Title.Enabled = false;
373                 t_Author.Enabled = false;
374                 c_Editor.Enabled = false;
375                 c_DocType.Enabled = false;
376                 t_source.Enabled = false;
377                 d_datetime.Enabled = false;
378                 l_ResumeOfBook.Enabled = false;
379                 t_BookDescription.Enabled = false;
380 
381                 b_Cancel.Visible = false;
382             }
383 
384             b_NewBook.Enabled = (mode ==
ControlsMode.SearchBooks || mode == ControlsMode.Navigate);
385             b_SaveBook.Enabled = (mode ==
ControlsMode.New_Book || mode == ControlsMode.Edit_Book);
386             b_DeleteBook.Enabled = ((functions.UserType ==
1) ? (current_book_id >= 0) : false);
387 
388             b_Previous.Enabled = (current_book_id > 0) &&
(mode == ControlsMode.SearchBooks || mode ==
ControlsMode.Navigate);
389             b_Next.Enabled = (current_book_id <
total_rows_found - 1) && (mode == ControlsMode.SearchBooks ||
mode == ControlsMode.Navigate);
390 
391             b_Upload.Enabled = (total_rows_found > 0) &&
(mode == ControlsMode.SearchBooks || mode ==
ControlsMode.Navigate);
392 
393             //Controlos indisponíveis para tblLogins
normais...
394             b_NewBook.Visible = ((functions.UserType == 1)
? true : false);
395             b_SaveBook.Visible = ((functions.UserType == 1)
? true : false);
396             g_CapaLivro.Visible = ((functions.UserType ==
1) ? true : false);
397             //btnManageUsers.Visible = ((Utils.UserType ==
1) ? true : false);
398             b_DeleteBook.Visible = ((functions.UserType ==
1) ? true : false);
399 
```

```
400                 b_DownloadFile.Enabled = (total_rows_found > 0)
&& (mode == ControlsMode.SearchBooks || mode ==
ControlsMode.Navigate) && hasBookPDF;
401                 b_UploadFile.Enabled = (total_rows_found > 0)
&& (mode == ControlsMode.SearchBooks || mode ==
ControlsMode.Navigate);
402                 b_UploadFile.Visible = ((functions.UserType ==
1) ? true : false);
403
404                 //Se não foi possível encontrar livros
405                 if (total_rows_found == 0)
406                 {
407                     //Mostrar que não foram encontrados
registos
408                     l_RecsFound.Text = "Livro 0 de 0";
409                 }
410                 else
411                 {
412                     //Se estamos no modo de criar novo livro,
mostrar que não foram encontrados registos
413                     if (mode == ControlsMode.New_Book)
414                         l_RecsFound.Text = "Livro 0 de 0";
415                     else //Mostrar quantos registos existem e
qual estamos a ver de momento
416                         l_RecsFound.Text = string.Format("Livro
{0} de {1}", current_book_id + 1, total_rows_found);
417                 }
418
419                 b_ManageEditors.Enabled = c_Editor.Enabled;
420
421                 b_ManageUsers.Enabled = (functions.UserType ==
1);
422             }
423
424             // Verificar se o controlo TextBox está vazio
425             private bool IsRTBEmpty(TextBox rtb)
426             {
427                 //string text = new
TextRange(rtb.Document.ContentStart,
rtb.Document.ContentEnd).Text;
428                 //return !String.IsNullOrWhiteSpace(text);
429
430                 return rtb.Text.Trim() == string.Empty;
431             }
432
```

```
433             // Métodos dos controlos
434             private void t_Reference_TextChanged(object sender,
EventArgs e)
435             {
436                 //Se o modo é diferente de novo livro
437                 if (mode != ControlsMode.New_Book)
438                     mode = ControlsMode.Edit_Book;   //Colocar
o modo como edição
439
440                 //Disponibilizar campos
441                 SetControls();
442             }
443             private void t_Title_TextChanged(object sender,
EventArgs e)
444             {
445                 //Se o modo é diferente de novo livro
446                 if (mode != ControlsMode.New_Book)
447                     mode = ControlsMode.Edit_Book;   //Colocar
o modo como edição
448
449                 //Disponibilizar campos
450                 SetControls();
451             }
452             private void t_Author_TextChanged(object sender,
EventArgs e)
453             {
454                 //Se o modo é diferente de novo livro
455                 if (mode != ControlsMode.New_Book)
456                     mode = ControlsMode.Edit_Book;   //Colocar
o modo como edição
457
458                 //Disponibilizar campos
459                 SetControls();
460             }
461             private void b_Search_Click(object sender,
EventArgs e)
462             {
463                 //Modo de pesquisa
464                 mode = ControlsMode.SearchBooks;
465
466                 //Carregamento dos livros baseados na pesquisa
efetuada
467                 FillComboOfBooksFound();
468
469                 //Disponibilizar campos
```

```
470                 SetControls();
471
472                 if (t_Search.ctl_TBox.Text.Trim() !=
string.Empty)
473                     l_RecsFound.Text += " (filtrado)";
474
475                 int doc_type_selected = ((KeyValuePair<string,
int>)c_DocType.ctl_CBox.SelectedItem).Value;
476                 //Disponibilizar se for um livro
477                 c_Editor.Enabled = (doc_type_selected == 1) &&
mode == ControlsMode.SearchBooks;
478                 t_source.Enabled = (doc_type_selected == 6) &&
mode == ControlsMode.SearchBooks;
479             }
480             private void b_NewBook_Click(object sender,
EventArgs e)
481             {
482                 t_Reference.ctl_TBox.Text = string.Empty;
483                 t_Title.ctl_TBox.Text = string.Empty;
484                 t_Author.ctl_TBox.Text = string.Empty;
485                 c_Books.ctl_CBox.SelectedIndex = -1;
486                 c_DocType.ctl_CBox.SelectedIndex = 0;
487                 t_BookDescription.Text = string.Empty;
488
489                 mode = ControlsMode.New_Book;
490
491                 //Disponibilizar campos
492                 SetControls();
493             }
494             private void b_Cancel_Click(object sender,
EventArgs e)
495             {
496                 if (mode == ControlsMode.New_Book)
497                     current_book_id = -1;
498                 if (c_Books.ctl_CBox.Items.Count > 0)
499                     c_Books.ctl_CBox.SelectedIndex =
current_book_id;
500                 loading_window = true;
501                 LoadCurrentSelectedBook();
502                 loading_window = false;
503                 mode = ControlsMode.SearchBooks;
504
505                 //Disponibilizar campos
506                 SetControls();
507             }
```

```
508         private void b_SaveBook_Click(object sender,
EventArgs e)
509         {
510             ControlsMode old_mode = mode;
511
512             try
513             {
514                 //Verificar se os campos estão preenchidos
515                 if (t_Reference.ctl_TBox.Text.Trim() ==
string.Empty ||
516                     t_Title.ctl_TBox.Text.Trim() ==
string.Empty ||
517                     t_Author.ctl_TBox.Text.Trim() ==
string.Empty ||
518                     c_Editor.ctl_CBox.SelectedIndex < 0 ||
519                     IsRTBEmpty(t_BookDescription) == true)
520                 {
521                     MessageBox.Show("Tem de preencher os
campos para criar um livro novo.", "Gravar Livro",
MessageBoxButtons.OK, MessageBoxIcon.Exclamation);
522                     return;
523                 }
524
525                 //Se tipo 'Livro' => Editora obrigatório
526                 if (((KeyValuePair<string,
int>)c_DocType.ctl_CBox.SelectedItem).Value == 1 &&
527                     c_Editor.ctl_CBox.SelectedIndex == 0)
528                 {
529                     MessageBox.Show("Para o tipo de
documento 'Livro' tem de escolher qual a editora.", "Gravar
Livro", MessageBoxButtons.OK, MessageBoxIcon.Exclamation);
530                     c_Editor.ctl_CBox.Focus();
531                     return;
532                 }
533
534                 //Validar tamanho do campo do resumo do
livro
535                 t_BookDescription.SelectAll();
536                 if (t_BookDescription.Text.Length > 2000)
537                 {
538                     MessageBox.Show("O resumo do livro não
pode conter mais do que 2000 caracteres.\r\nTotal: " +
t_BookDescription.Text.Length.ToString() + " caracteres
encontrados.\r\n\r\nReduza o texto por forma a poder gravar",
```

```
"Gravar Livro", MessageBoxButtons.OK,
MessageBoxIcon.Exclamation);
539                         return;
540                     }
541
542                     //Se o modo é a criação de um livro novo
543                     if (mode == ControlsMode.New_Book)
544                     {
545                         //Liga à base de dados com um
TableAdapter
546
DataSetsTableAdapters.tblBooksTableAdapter tbta = new
DataSetsTableAdapters.tblBooksTableAdapter();
547                         tbta.Connection.ConnectionString =
functions.Get_DBConnection();
548                         //Carregar a tabela
549                         DataSets.tblBooksDataTable tbdt =
tbta.GetData_tblBooks(null, null, null, null, null, null, null,
null, null);
550
551                         int ret = -1;
552                         if (tbdt != null)
553                         {
554                             string reference = null;
555                             reference =
t_Reference.ctl_TBox.Text;
556
557                             //Liga à base de dados com um
TableAdapter
558
DataSetsTableAdapters.tblBooksTableAdapter tbta_2 = new
DataSetsTableAdapters.tblBooksTableAdapter();
559                             tbta_2.Connection.ConnectionString
= functions.Get_DBConnection();
560                             //Carregar a tabela
561                             DataSets.tblBooksDataTable tbdt_2 =
tbta_2.GetData_tblBooks(null, reference, null, null, null, null,
null, null, null);
562                             if (tbdt_2 != null &&
tbdt_2.Rows.Count >= 1)
563                             {
564                                 MessageBox.Show("Erro a
adicionar o Livro.\r\nA referência '" +
t_Reference.ctl_TBox.Text + "' já existe.", "Erro",
MessageBoxButtons.OK, MessageBoxIcon.Exclamation);
```

```
565                             return;
566                         }
567
568                         DataSets.tblBooksRow rowToSave =
null;
569                         //Se fôr no modo de novo livro
adicionar uma nova 'row'
570                         if (mode == ControlsMode.New_Book)
571                         {
572                             rowToSave =
tbdt.NewtblBooksRow();
573                         }
574
575                         //Atribuir os valores aos campos da
'row'
576                         rowToSave.book_code =
t_Reference.ctl_TBox.Text;
577                         rowToSave.book_name =
t_Title.ctl_TBox.Text;
578                         rowToSave.author =
t_Author.ctl_TBox.Text;
579                         int editor_id =
((KeyValuePair<string,
int>)c_Editor.ctl_CBox.SelectedItem).Value;
580                         int doc_type =
((KeyValuePair<string,
int>)c_DocType.ctl_CBox.SelectedItem).Value;
581                         if (editor_id != -1)
582                             rowToSave.editor = editor_id;
583                         else
584                             rowToSave.SeteditorNull();
585                         rowToSave.document_type = doc_type;
586                         if (doc_type == 6)
587                             rowToSave.document_source =
t_source.ctl_TBox.Text;
588                         else
589                             rowToSave.document_source =
string.Empty;
590                         rowToSave.document_date =
d_datetime.ctl_DTime.Value;
591                         t_BookDescription.SelectAll();
592                         rowToSave.resume =
t_BookDescription.Text;
593
```

```
594                             //Se fôr no modo de novo livro
adicionar a 'row' à tabela
595                             if (mode == ControlsMode.New_Book)
596                                 tbdt.Rows.Add(rowToSave);
597
598                             try
599                             {
600                                 //Criar/Atualizar o registo
601                                 ret = tbta.Update(rowToSave);
602                                 if (ret == 1)
603                                 {
604                                     //Adicionado com sucesso!
605                                     FillComboOfBooksFound();
606                                 }
607                             }
608                             catch (Exception ex)
609                             {
610                                 MessageBox.Show("Erro: " +
ex.Message, "Erro", MessageBoxButtons.OK, MessageBoxIcon.Error);
611                             }
612                         }
613                     }
614
615                     //Se o modo é a alteração de um livro
existente
616                     if (mode == ControlsMode.Edit_Book)
617                     {
618                         //Obter o ID do livro selecionado
619                         int book_id = ((KeyValuePair<string,
int>)c_Books.ctl_CBox.SelectedItem).Value;
620
621                         //Liga à base de dados com um
TableAdapter
622
DataSetsTableAdapters.tblBooksTableAdapter tbta = new
DataSetsTableAdapters.tblBooksTableAdapter();
623                         tbta.Connection.ConnectionString =
functions.Get_DBConnection();
624                         //Carregar a tabela
625                         DataSets.tblBooksDataTable tbdt =
tbta.GetData_tblBooks(book_id, null, null, null, null, null,
null, null, null);
626
627                         int ret = -1;
628                         if (tbdt != null)
```

```
629                     {
630                         //Liga à base de dados com um
TableAdapter
631
DataSetsTableAdapters.tblBooksTableAdapter tbta_2 = new
DataSetsTableAdapters.tblBooksTableAdapter();
632                         tbta_2.Connection.ConnectionString
= functions.Get_DBConnection();
633                         //Carregar a tabela
634                         DataSets.tblBooksDataTable tbdt_2 =
tbta_2.GetData_tblBooks(null, t_Reference.ctl_TBox.Text, null,
null, null, null, null, null, null);
635
636                         //Se encontrou o registo
637                         if (tbdt_2 != null &&
tbdt_2.Rows.Count >= 1)
638                         {
639                             DataSets.tblBooksRow rowToEdit
= ((DataSets.tblBooksRow)tbdt_2.Rows[0]);
640
641                             //Ver se a referência foi
encontrada noutro livro
642                             if (rowToEdit.book_code ==
t_Reference.ctl_Label.Text && rowToEdit.id != book_id)
643                             {
644                                 MessageBox.Show("Erro a
alterar.\r\nA referência '" + t_Reference.ctl_TBox.Text + "' já
existe.", "Erro", MessageBoxButtons.OK,
MessageBoxIcon.Exclamation);
645                                 return;
646                             }
647                         }
648
649                         DataSets.tblBooksRow rowToSave =
null;
650
651                         //Modo de criação?
652                         if (mode == ControlsMode.New_Book)
653                         {
654                             //Novo registo
655                             rowToSave =
tbdt.NewtblBooksRow();
656                         }
657                         //Modo de alteração?
658                         if (mode == ControlsMode.Edit_Book)
```

```
659                             {
660                                 //Obter o registo
661                                 rowToSave =
tbdt.FindByid(book_id);
662                             }
663
664                             //Por os valores nos campos
665                             rowToSave.book_code =
t_Reference.ctl_TBox.Text;
666                             rowToSave.book_name =
t_Title.ctl_TBox.Text;
667                             rowToSave.author =
t_Author.ctl_TBox.Text;
668                             int editor_id =
((KeyValuePair<string,
int>)c_Editor.ctl_CBox.SelectedItem).Value;
669                             int doc_type =
((KeyValuePair<string,
int>)c_DocType.ctl_CBox.SelectedItem).Value;
670                             if (editor_id != -1)
671                                 rowToSave.editor = editor_id;
672                             else
673                                 rowToSave.SeteditorNull();
674                             rowToSave.document_type = doc_type;
675                             if (doc_type == 6)
676                                 rowToSave.document_source =
t_source.ctl_TBox.Text;
677                             else
678                                 rowToSave.document_source =
string.Empty;
679                             rowToSave.document_date =
d_datetime.ctl_DTime.Value;
680                             t_BookDescription.SelectAll();
681                             rowToSave.resume =
t_BookDescription.Text;
682
683                             //Modo de criação?
684                             if (mode == ControlsMode.New_Book)
685                             {
686                                 //Adicionar a 'row' à tabela
687                                 tbdt.Rows.Add(rowToSave);
688                             }
689
690                             try
691                             {
```

```
692                             //Atualizar a base de dados
693                             ret = tbta.Update(rowToSave);
694                             if (ret == 1)
695                             {
696                                 //Atualizado
697                                 FillComboOfBooksFound();
698                             }
699                         }
700                         catch (Exception ex)
701                         {
702                             MessageBox.Show("Erro: " +
ex.Message, "Erro", MessageBoxButtons.OK, MessageBoxIcon.Error);
703                         }
704                     }
705                 }
706
707                 //Limpar os controlos
708                 t_Reference.ctl_TBox.Text = string.Empty;
709                 t_Title.ctl_TBox.Text = string.Empty;
710                 t_Author.ctl_TBox.Text = string.Empty;
711                 c_Editor.ctl_CBox.SelectedIndex = 0;
712                 c_DocType.ctl_CBox.SelectedIndex = 0;
713                 t_BookDescription.Text = string.Empty;
714
715                 //Modo de criação?
716                 if (mode == ControlsMode.New_Book)
717                     current_book_id = 0;
718                 c_Books.ctl_CBox.SelectedIndex =
current_book_id;
719
720                 //Carregar o livro selecionado
721                 loading_window = true;
722                 LoadCurrentSelectedBook();
723                 loading_window = false;
724
725                 //Disponibilizar campos
726                 SetControls();
727
728                 //Modo de pesquisa
729                 mode = ControlsMode.SearchBooks;
730             }
731             catch (Exception ex)
732             {
733                 MessageBox.Show("Erro: " + ex.Message,
"Erro", MessageBoxButtons.OK, MessageBoxIcon.Error);
```

```
734                     return;
735                 }
736
737                 int old_current_book_id = current_book_id;
738                 //Carregamento dos livros baseados na pesquisa
efetuada
739                 FillComboOfBooksFound();
740
741                 current_book_id = old_current_book_id;
742
743                 if (old_mode == ControlsMode.Edit_Book)
744                     c_Books.ctl_CBox.SelectedIndex =
current_book_id;
745
746                 int doc_type_selected = ((KeyValuePair<string,
int>)c_DocType.ctl_CBox.SelectedItem).Value;
747                 //Disponibilizar se for um livro
748                 c_Editor.Enabled = (doc_type_selected == 1) &&
mode == ControlsMode.SearchBooks;
749                 t_source.Enabled = (doc_type_selected == 6) &&
mode == ControlsMode.SearchBooks;
750             }
751             private void b_Previous_Click(object sender,
EventArgs e)
752             {
753                 //Modo de navegação
754                 mode = ControlsMode.Navigate;
755
756                 //Navegar para trás
757                 current_book_id--;
758                 c_Books.ctl_CBox.SelectedIndex =
current_book_id;
759
760                 //Disponibilizar campos
761                 SetControls();
762             }
763             private void b_Next_Click(object sender, EventArgs
e)
764             {
765                 //Modo de navegação
766                 mode = ControlsMode.Navigate;
767
768                 //Navegar para a frente
769                 current_book_id++;
```

```
770                 c_Books.ctl_CBox.SelectedIndex =
current_book_id;
771
772                 //Disponibilizar campos
773                 SetControls();
774             }
775             private void b_Upload_Click(object sender,
EventArgs e)
776             {
777                 //Obter os bytes de uma imagem
778                 byte[] picbyte = functions.OpenFile(null,
"Image Files (*.bmp; *.jpg; *.png; *.gif)|*.bmp; *.jpg; *.png;
*.gif", "Escolher uma imagem para o livro");
779
780                 if (picbyte != null)
781                 {
782                     //Obter o ID do livro selecionado
783                     int book_id = ((KeyValuePair<string,
int>)c_Books.ctl_CBox.SelectedItem).Value;
784
785                     //Liga à base de dados com um TableAdapter
786                     DataSetsTableAdapters.tblBooksTableAdapter
tbta = new DataSetsTableAdapters.tblBooksTableAdapter();
787                     tbta.Connection.ConnectionString =
functions.Get_DBConnection();
788                     //Carregar a tabela
789                     DataSets.tblBooksDataTable tbdt =
tbta.GetData_tblBooks(book_id, null, null, null, null, null,
null, null, null);
790
791                     if (tbdt != null && tbdt.Rows.Count > 0)
792                     {
793                         DataSets.tblBooksRow row =
((DataSets.tblBooksRow)tbdt.Rows[0]);
794
795                         //Se o registo foi encontrado
796                         if (row != null)
797                         {
798                             //Atribuir a imagem ao campo
799                             row.login_image = picbyte;
800
801                             //Atualizar a pré-visualização do
livro
802                             int ret = tbta.Update(row);
803                             if (ret == 1)
```

```
804                             {
805                                 Image img =
functions.GetImage_ByteArray(picbyte);
806
807                                 //Colocar a imagem no controlo
808                                 i_BookCover.Image = img;
809                             }
810                         }
811                     }
812                 }
813
814                 //Libertar memória
815                 picbyte = null;
816             }
817             private void b_ManageUsers_Click(object sender,
EventArgs e)
818             {
819                 //Abrir a janela de manutenção de utilizadores
820                 ManageUsers manUsers = new ManageUsers();
821                 manUsers.Owner = this;
822                 manUsers.ShowDialog();
823
824                 LoginWithCurrentUser();
825
826                 SetControls();
827             }
828             private void b_DownloadPDF_Click(object sender,
EventArgs e)
829             {
830                 //Se não existe nenhum livro selecionado sair
da função
831                 if (c_Books.ctl_CBox.SelectedItem == null)
832                     return;
833
834                 //Obter o ID do livro selecionado
835                 int book_id = ((KeyValuePair<string,
int>)c_Books.ctl_CBox.SelectedItem).Value;
836
837                 //Liga à base de dados com um TableAdapter
838                 DataSetsTableAdapters.tblBooksTableAdapter tbta
= new DataSetsTableAdapters.tblBooksTableAdapter();
839                 tbta.Connection.ConnectionString =
functions.Get_DBConnection();
840                 //Carregar a tabela
```

```
841             DataSets.tblBooksDataTable tbdt =
tbta.GetData_tblBooks(book_id, null, null, null, null, null,
null, null, null);
842
843             //Se encontrou um registo
844             if (tbdt != null && tbdt.Rows.Count > 0)
845             {
846                 DataSets.tblBooksRow row =
((DataSets.tblBooksRow)tbdt.Rows[0]);
847
848                 //Se tem um ficheiro em PDF
849                 if (row.IsfileNull() == false)
850                 {
851                     //Pedir para gravar o PDF onde quiser
852                     SaveFileDialog sfd = new
SaveFileDialog();
853                     sfd.CheckPathExists = true;
854                     sfd.Filter = functions.filtro;
855                     sfd.Title = "Onde deseja guardar o
livro?";
856                     sfd.OverwritePrompt = true;
857
858                     //Atribuir uma sugestão de nome para a
gravação do ficheiro
859                     string filename = row.book_name + " - "
+ row.author + (row.IseditorNull() ? string.Empty : " - " +
functions.GetDesc_Editor(row.editor));
860                     filename =
functions.FixFilenameChars(filename);
861                     sfd.FileName = filename;
862                     if (sfd.ShowDialog() ==
DialogResult.OK)
863                     {
864                         if
(functions.SaveByteArray_File(sfd.FileName, row.file))
865                         {
866                             //Ficheiro guardado
867                         }
868                     }
869                 }
870             }
871         }
872         private void b_UploadPDF_Click(object sender,
EventArgs e)
873         {
```

```
874                 //Ler o ficheiro obtido do disco
875                 byte[] pdfbyte = functions.OpenFile(null,
functions.filtro, "Escolher um ficheiro do livro");
876
877                 //Se o ficheiro foi lido
878                 if (pdfbyte != null)
879                 {
880                     //Obter o ID do livro selecionado
881                     int book_id = ((KeyValuePair<string,
int>)c_Books.ctl_CBox.SelectedItem).Value;
882
883                     //Liga à base de dados com um TableAdapter
884                     DataSetsTableAdapters.tblBooksTableAdapter
tbta = new DataSetsTableAdapters.tblBooksTableAdapter();
885                     tbta.Connection.ConnectionString =
functions.Get_DBConnection();
886                     //Carregar a tabela
887                     DataSets.tblBooksDataTable tbdt =
tbta.GetData_tblBooks(book_id, null, null, null, null, null,
null, null, null);
888
889                     //Se encontrou o registo
890                     if (tbdt != null && tbdt.Rows.Count > 0)
891                     {
892                         DataSets.tblBooksRow row =
((DataSets.tblBooksRow)tbdt.Rows[0]);
893                         if (row != null)
894                         {
895                             //Atribuir o ficheiro PDF ao campo
896                             row.file = pdfbyte;
897
898                             //Atualizar a base de dados
899                             int ret = tbta.Update(row);
900                             if (ret == 1)
901                             {
902                                 //Existe um PDF
903                                 hasBookPDF = true;
904                                 //Disponibilizar o botão de
Download do PDF
905                                 b_DownloadFile.Enabled = true;
906                             }
907                         }
908                     }
909                 }
910
```

```
911                 //Libertar memória
912                 pdfbyte = null;
913             }
914             private void b_ManageEditors_Click(object sender,
EventArgs e)
915             {
916                 //Abrir a janela de gestão de tblEditors
917                 ManageEditors manPubs = new ManageEditors();
918                 manPubs.Owner = this;
919                 manPubs.ShowDialog();
920 
921                 //Recarregar as editoras
922                 LoadEditors();
923             }
924             private void b_DeleteBook_Click(object sender,
EventArgs e)
925             {
926                 int book_id = -1;
927 
928                 if (c_Books.ctl_CBox.SelectedItem != null)
929                 {
930                     book_id = ((KeyValuePair<string,
int>)c_Books.ctl_CBox.SelectedItem).Value;
931 
932                     //Liga à base de dados com um TableAdapter
933                     DataSetsTableAdapters.tblBooksTableAdapter
tab_adap = new DataSetsTableAdapters.tblBooksTableAdapter();
934                     tab_adap.Connection.ConnectionString =
functions.Get_DBConnection();
935                     //Carregar a tabela
936                     DataSets.tblBooksDataTable dtab =
tab_adap.GetData_tblBooks(book_id, null, null, null, null, null,
null, null, null);
937 
938                     //Se a editora não existe
939                     if (dtab.Rows.Count == 0)
940                     {
941                         MessageBox.Show("O livro não existe na
base de dados.", "Livro não existe", MessageBoxButtons.OK,
MessageBoxIcon.Exclamation);
942                         return;
943                     }
944 
945                     //Obter o registo
```

```
946                     DataSets.tblBooksRow row =
((DataSets.tblBooksRow)dtab.Rows[0]);
947
948                     if (row != null)
949                     {
950                         DialogResult resp =
MessageBox.Show("Deseja remover o livro?", "Remover Livro",
MessageBoxButtons.YesNo, MessageBoxIcon.Question,
MessageBoxDefaultButton.Button2);
951
952                         if (resp == DialogResult.Yes)
953                         {
954                             //Eliminar o registo
955                             row.Delete();
956
957                             //Atualizar a base de dados
958                             int ret = tab_adap.Update(row);
959                             if (ret > 0)
960                             {
961                                 MessageBox.Show("Livro
removido", "Remover Livro", MessageBoxButtons.OK,
MessageBoxIcon.Information);
962
963                                 //Remover o item da combo
964
c_Books.ctl_CBox.Items.RemoveAt(c_Books.ctl_CBox.SelectedIndex);
965
966                                 //Carregar o livro atualmente
selecionado
967                                 LoadCurrentSelectedBook();
968
969                                 //Carregar novamente a
pesquisa, simulando o click do botão 'Pesquisar'
970                                 b_Search_Click(b_Search, new
EventArgs());
971                             }
972                         }
973                     }
974                 }
975             }
976             private void t_BookDescription_TextChanged(object
sender, EventArgs e)
977             {
978                 //Não é modo de criação?
979                 if (mode != ControlsMode.New_Book)
```

```
980                     mode = ControlsMode.Edit_Book;
981
982                 //Disponibilizar campos
983                 SetControls();
984             }
985             private void c_Editor_SelectedIndexChanged(object
sender, EventArgs e)
986             {
987                 //Não é modo de criação?
988                 if (mode != ControlsMode.New_Book)
989                     mode = ControlsMode.Edit_Book;
990
991                 //Disponibilizar campos
992                 SetControls();
993             }
994             private void
c_DocType_ctl_CBox_SelectedIndexChanged(object sender, EventArgs
e)
995             {
996                 //Não é modo de criação?
997                 if (mode != ControlsMode.New_Book)
998                     mode = ControlsMode.Edit_Book;
999
1000                int doc_type_selected = ((KeyValuePair<string,
int>)c_DocType.ctl_CBox.SelectedItem).Value;
1001
1002                if (doc_type_selected != 1)
1003                    c_Editor.ctl_CBox.SelectedIndex = 0;
1004                if (doc_type_selected != 6)
1005                    t_source.ctl_TBox.Text = string.Empty;
1006
1007                //Disponibilizar se for um livro
1008                c_Editor.Enabled = (doc_type_selected == 1) &&
mode != ControlsMode.SearchBooks;
1009                t_source.Enabled = (doc_type_selected == 6) &&
mode != ControlsMode.SearchBooks;
1010
1011                b_ManageEditors.Enabled = c_Editor.Enabled;
1012
1013                SetControls();
1014            }
1015            private void c_Books_SelectionChanged(object
sender, EventArgs e)
1016            {
1017                //Obter o índice do item da combo dos livros
```

```
1018                current_book_id =
c_Books.ctl_CBox.SelectedIndex;
1019
1020                //Carregar o livro selecionado
1021                LoadCurrentSelectedBook();
1022
1023                //Disponibilizar campos
1024                SetControls();
1025            }
1026        }
1027    }
```

Anexo III – ManageEditors.cs

```
1     using System;
2     using System.Collections.Generic;
3     using System.ComponentModel;
4     using System.Data;
5     using System.Drawing;
6     using System.Linq;
7     using System.Text;
8     using System.Windows.Forms;
9
10    namespace VirtualLibrary
11    {
12        public partial class ManageEditors : Form
13        {
14            public ManageEditors()
15            {
16                InitializeComponent();
17            }
18
19            // carregamento da janela
```

```
20          private void ManageEditors_Load(object sender,
EventArgs e)
21          {
22              //Carregar as editoras
23              FillEditors();
24          }
25
26          // Carregamento das editoras
27          private void FillEditors()
28          {
29              //Liga à base de dados com um TableAdapter
30              DataSetsTableAdapters.tblEditorsTableAdapter
ta_logins = new DataSetsTableAdapters.tblEditorsTableAdapter();
31              ta_logins.Connection.ConnectionString =
functions.Get_DBConnection();
32              //Carregar a tabela
33              DataSets.tblEditorsDataTable dt_logins =
ta_logins.GetData_tblEditors(null, null);
34
35              //Limpar a datagrid dos utilizadores
36              dg_Editors.Rows.Clear();
37
38              //Percorrer os registos
39              foreach (DataSets.tblEditorsRow row in
dt_logins.Rows)
40              {
41                  //carregar uma class de apoio à criação do
registo para a datagrid
42                  var new_editor = new Editor
43                  {
44                      ID = Convert.ToInt32(row.ItemArray[0]),
45                      Name = row.ItemArray[1].ToString()
46                  };
47
```

```
48                  //adicionar o registo à datagrid
49                  //assuming that you created columns (via
code or designer) in myDGV
50                  DataGridViewRow new_row =
(DataGridViewRow)dg_Editors.RowTemplate.Clone();
51                  object[] cells = {
new_editor.ID.ToString(), new_editor.Name };
52                  new_row.CreateCells(dg_Editors, cells);
53                  new_row.Tag = new_editor;
54                  try
55                  {
56                      dg_Editors.Rows.Add(new_row);
57                  }
58                  catch (Exception ex)
59                  {
60                      MessageBox.Show(ex.Message);
61                  }
62              }
63
64              dg_Editors.ClearSelection();
65          }
66
67          // Limpar os controlos
68          private void cleanControls()
69          {
70              t_ID.ctl_TBox.Text = string.Empty;
71              t_Editor.ctl_TBox.Text = string.Empty;
72          }
73          private void LinhaSelecionada(DataGridViewRow
dgv_row)
74          {
75              //Se não existe nenhuma linha selecionada
76              if (dgv_row == null)
77              {
```

```
78                  //Limpar os controlos
79                  cleanControls();
80
81                  //Desselecionar a linha da datagrid
82                  dg_Editors.ClearSelection();
83              }
84
85              //Disponibilizar ou não os controlos de alterar
e remover caso esteja ou não selecionada
86              b_Change.Enabled = (dgv_row != null);
87              b_Remove.Enabled = (dgv_row != null);
88          }
89
90          // Métodos dos controlos
91          private void dg_Editors_SelectionChanged(object
sender, EventArgs e)
92          {
93              //Limpar os controlos
94              cleanControls();
95
96              //Se existe uma linha selecionada
97              if (dg_Editors.SelectedRows.Count > 0)
98              {
99                  //Obter o utilizador selecionado
100                 var editor_selected =
((Editor)dg_Editors.SelectedRows[0].Tag);
101
102                 if (editor_selected != null)
103                 {
104                     //Atribuir os valores a cada controlo
105                     t_ID.ctl_TBox.Text =
editor_selected.ID.ToString();
106                     t_Editor.ctl_TBox.Text =
editor_selected.Name;
```

```
107                 }
108             }
109         }
110         private void dg_Editors_MouseUp(object sender,
MouseEventArgs e)
111         {
112             //Validar se o utilizador clicou numa linha com
o rato
113             DataGridView.HitTestInfo htinfo =
dg_Editors.HitTest(e.X, e.Y);
114
115             DataGridViewRow row = null;
116
117             if (htinfo.RowIndex >= 0 && htinfo.ColumnIndex
>= 0)
118                 row = dg_Editors.Rows[htinfo.RowIndex];
119
120             //Código para quando se seleciona ou não uma
linha da datagrid
121             LinhaSelecionada(row);
122         }
123         private void b_Add_Click(object sender, EventArgs
e)
124         {
125             //Verificar os campos de introdução obrigatória
126             if (t_Editor.ctl_TBox.Text.Trim() ==
string.Empty)
127             {
128                 MessageBox.Show("Os campos são
obrigatórios.", "Campos obrigatórios", MessageBoxButtons.OK,
MessageBoxIcon.Exclamation);
129                 return;
130             }
131
132             //Liga à base de dados com um TableAdapter
```

```
133             DataSetsTableAdapters.tblEditorsTableAdapter
ta_logins = new DataSetsTableAdapters.tblEditorsTableAdapter();
134             ta_logins.Connection.ConnectionString =
functions.Get_DBConnection();
135             //Carregar a tabela
136             DataSets.tblEditorsDataTable dt_logins =
ta_logins.GetData_tblEditors(null, t_Editor.Text);
137
138             //Se encontrou algum registo
139             if (dt_logins.Rows.Count > 0)
140             {
141                 MessageBox.Show("A editora a adicionar já
existe na base de dados.", "Editora já existe",
MessageBoxButtons.OK, MessageBoxIcon.Exclamation);
142                 return;
143             }
144
145             //Criar um novo registo
146             DataSets.tblEditorsRow row =
dt_logins.NewtblEditorsRow();
147
148             //Atribuir os valores aos campos do registo
149             row.editor_name = t_Editor.ctl_TBox.Text;
150
151             //Adicionar o registo à tabela
152             dt_logins.Rows.Add(row);
153
154             //Atualizar a base de dados
155             int ret = ta_logins.Update(dt_logins);
156             if (ret > 0)
157             {
158                 //Carregar os utilizadores
159                 FillEditors();
160             }
```

```
161         }
162         private void b_Change_Click(object sender,
EventArgs e)
163         {
164             //Liga à base de dados com um TableAdapter
165             DataSetsTableAdapters.tblEditorsTableAdapter
ta_logins = new DataSetsTableAdapters.tblEditorsTableAdapter();
166             ta_logins.Connection.ConnectionString =
functions.Get_DBConnection();
167             //Carregar a tabela
168             DataSets.tblEditorsDataTable dt_logins =
ta_logins.GetData_tblEditors(Convert.ToInt32(t_ID.ctl_TBox.Text)
, null);
169
170             //Se o utilizador não existe
171             if (dt_logins.Rows.Count == 0)
172             {
173                 MessageBox.Show("A editora a alterar não
existe na base de dados.", "Editora não existe",
MessageBoxButtons.OK, MessageBoxIcon.Exclamation);
174                 return;
175             }
176
177             //Obter o registo
178             DataSets.tblEditorsRow row =
((DataSets.tblEditorsRow)dt_logins.Rows[0]);
179
180             if (row != null)
181             {
182                 //Atribuir os valores aos campos do registo
183                 row.editor_name = t_Editor.ctl_TBox.Text;
184
185                 //Atualizar a base de dados
186                 int ret = ta_logins.Update(row);
187                 if (ret > 0)
```

```
188                 {
189                     //Carregar os utilizadores
190                     FillEditors();
191                 }
192             }
193         }
194         private void b_Remove_Click(object sender,
EventArgs e)
195         {
196             //Ligação à base de dados através de um
TableAdapter
197             DataSetsTableAdapters.tblBooksTableAdapter tbta
= new DataSetsTableAdapters.tblBooksTableAdapter();
198             tbta.Connection.ConnectionString =
functions.Get_DBConnection();
199             //Carregamento da tabela
200             DataSets.tblBooksDataTable tbdt =
tbta.GetData_tblBooks(null, null, null, null,
Convert.ToInt32(t_ID.ctl_TBox.Text), null, null, null, null);
201
202             //Está a ser utilizada?
203             if (tbdt.Rows.Count > 0)
204             {
205                 //Não deixar eliminar
206                 MessageBox.Show("A editora a remover não
pode ser removida porque está a ser utilizada.", "Remover
Editora", MessageBoxButtons.OK, MessageBoxIcon.Exclamation);
207                 return;
208             }
209
210             //Liga à base de dados com um TableAdapter
211             DataSetsTableAdapters.tblEditorsTableAdapter
ta_logins = new DataSetsTableAdapters.tblEditorsTableAdapter();
212             ta_logins.Connection.ConnectionString =
functions.Get_DBConnection();
```

```
213                 //Carregar a tabela
214                 DataSets.tblEditorsDataTable dt_logins =
ta_logins.GetData_tblEditors(Convert.ToInt32(t_ID.ctl_TBox.Text)
, null);
215
216                 //Se o utilizador não existe
217                 if (dt_logins.Rows.Count == 0)
218                 {
219                     MessageBox.Show("A editora a remover não
existe na base de dados.", "Editora não existe",
MessageBoxButtons.OK, MessageBoxIcon.Exclamation);
220                     return;
221                 }
222
223                 //Obter o registo
224                 DataSets.tblEditorsRow row =
((DataSets.tblEditorsRow)dt_logins.Rows[0]);
225
226                 if (row != null)
227                 {
228                     DialogResult resp = MessageBox.Show("Deseja
remover a editora selecionada?", "Remover Editora",
MessageBoxButtons.YesNo, MessageBoxIcon.Question,
MessageBoxDefaultButton.Button2);
229
230                     if (resp == DialogResult.Yes)
231                     {
232                         //Eliminar o registo
233                         row.Delete();
234
235                         //Atualizar a base de dados
236                         int ret = ta_logins.Update(row);
237                         if (ret > 0)
238                         {
239                             //Carregar os utilizadores
```

```
240                         FillEditors();
241                     }
242                 }
243             }
244         }
245     }
246
247     // Class de editoras
248     public class Editor
249     {
250         private int _ID;
251         private string _Name;
252
253         public int ID
254         {
255             get { return _ID; }
256             set { _ID = value; }
257         }
258         public string Name
259         {
260             get { return _Name; }
261             set { _Name = value; }
262         }
263
264         // Construtor da Class
265         public Editor()
266         {
267
268         }
269
270         // Inicialização de variáveis
271         public Editor(int id, string name)
```

```
272             {
273                 this.ID = id;
274                 this.Name = name;
275             }
276         }
277     }
```

Anexo – IV ManageUsers.cs

```
1     using System;
2     using System.Collections.Generic;
3     using System.ComponentModel;
4     using System.Data;
5     using System.Drawing;
6     using System.Linq;
7     using System.Text;
8     using System.Windows.Forms;
9
10    namespace VirtualLibrary
11    {
12        public partial class ManageUsers : Form
13        {
14            public ManageUsers()
15            {
16                InitializeComponent();
17            }
18
19            private void ManageUsers_Load(object sender,
EventArgs e)
20            {
21                //Carregar os tipos de utilizadores
22                FillUserTypes();
23
24                //Carregar os utilizadores
25                FillUsers();
26            }
27
28            // Carregamento dos utilizadores
```

```
29          private void FillUsers()
30          {
31              //Liga à base de dados com um TableAdapter
32              DataSetsTableAdapters.tblLoginsTableAdapter
ta_logins = new DataSetsTableAdapters.tblLoginsTableAdapter();
33              ta_logins.Connection.ConnectionString =
functions.Get_DBConnection();
34              //Carregar a tabela
35              DataSets.tblLoginsDataTable dt_logins =
ta_logins.GetData_tblLogins(null, null, null);
36
37              //Limpar a datagrid dos utilizadores
38              dgUsers.Rows.Clear();
39
40              //Percorrer os registos
41              foreach (DataSets.tblLoginsRow row in
dt_logins.Rows)
42              {
43                  //carregar uma class de apoio à criação do
registo para a datagrid
44                  var new_user = new User
45                  {
46                      ID = Convert.ToInt32(row.ItemArray[0]),
47                      Username = row.ItemArray[1].ToString(),
48                      UserType =
Convert.ToInt32(row.ItemArray[2]),
49                      UserTypeDesc =
functions.GetDesc_UserType(Convert.ToInt32(row.ItemArray[2]))
50                  };
51
52                  //adicionar o registo à datagrid
53                  //assuming that you created columns (via
code or designer) in myDGV
54                  DataGridViewRow new_row =
(DataGridViewRow)dgUsers.RowTemplate.Clone();
55                  object[] cells = { new_user.ID.ToString(),
new_user.Username, new_user.UserTypeDesc };
56                  new_row.CreateCells(dgUsers, cells);
57                  new_row.Tag = new_user;
58                  try
59                  {
60                      dgUsers.Rows.Add(new_row);
61                  }
62                  catch (Exception ex)
63                  {
```

```
64                          MessageBox.Show(ex.Message);
65                      }
66                  }
67
68                  dgUsers.ClearSelection();
69              }
70
71              // Carregamento dos tipos de utilizadores
72              private void FillUserTypes()
73              {
74                  //Liga à base de dados com um TableAdapter
75                  DataSetsTableAdapters.tblLoginTypesTableAdapter
tpta = new DataSetsTableAdapters.tblLoginTypesTableAdapter();
76                  tpta.Connection.ConnectionString =
functions.Get_DBConnection();
77                  //Carregar a tabela
78                  DataSets.tblLoginTypesDataTable tpdt =
tpta.GetData_tblLoginTypes(null);
79
80                  //Limpar a datagrid
81                  cmbUsertype.ctl_CBox.Items.Clear();
82
83                  //Carregamento da combo
84                  cmbUsertype.ctl_CBox.DisplayMember = "Key";
85                  cmbUsertype.ctl_CBox.ValueMember = "Value";
86                  cmbUsertype.ctl_CBox.Items.Add(new
KeyValuePair<string, int>(string.Empty, 0));
87                  foreach (DataSets.tblLoginTypesRow row in
tpdt.Rows)
88                      cmbUsertype.ctl_CBox.Items.Add(new
KeyValuePair<string, int>(row.type, row.id));
89
90                  //Se houver pelo menos 1 registo, selecioná-lo
91                  if (cmbUsertype.ctl_CBox.Items.Count > 0)
cmbUsertype.ctl_CBox.SelectedIndex = 0;
92              }
93
94              // Limpar os controlos
95              private void cleanControls()
96              {
97                  txtID.ctl_TBox.Text = string.Empty;
98                  txtUsername.ctl_TBox.Text = string.Empty;
99                  functions.SelectComboItem(cmbUsertype.ctl_CBox,
0);
100             }
```

```
101         private void LinhaSelecionada(DataGridViewRow
targetRow)
102         {
103             //Se não existe nenhuma linha selecionada
104             if (targetRow == null)
105             {
106                 //Limpar os controlos
107                 cleanControls();
108 
109                 //Desselecionar a linha da datagrid
110                 dgUsers.ClearSelection();
111             }
112 
113             //Disponibilizar ou não os controlos de alterar
e remover caso esteja ou não selecionada
114             btnChange.Enabled = (targetRow != null);
115             btnRemove.Enabled = (targetRow != null);
116 
117             // Se for o admin do VMM
118             if (dgUsers.SelectedRows.Count == 1)
119             {
120                 var user_selected =
((User)dgUsers.SelectedRows[0].Tag);
121                 if (user_selected.ID == 1)
122                 {
123                     btnChange.Enabled = false;
124                     btnRemove.Enabled = false;
125                 }
126             }
127         }
128 
129         // Controlos
130         private void dgUsers_SelectionChanged(object
sender, EventArgs e)
131         {
132             //Limpar os controlos
133             cleanControls();
134 
135             //Se existe uma linha selecionada
136             if (dgUsers.SelectedRows.Count > 0)
137             {
138                 //Obter o utilizador selecionado
139                 var user_selected =
((User)dgUsers.SelectedRows[0].Tag);
140 
```

```
141                     if (user_selected != null)
142                     {
143                         //Atribuir os valores a cada controlo
144                         txtID.ctl_TBox.Text =
user_selected.ID.ToString();
145                         txtUsername.ctl_TBox.Text =
user_selected.Username;
146
functions.SelectComboItem(cmbUsertype.ctl_CBox,
user_selected.UserType);
147                     }
148                     LinhaSelecionada(dgUsers.SelectedRows[0]);
149                 }
150             }
151             private void dgUsers_MouseUp(object sender,
MouseEventArgs e)
152             {
153                 //Validar se o utilizador clicou numa linha com
o rato
154                 DataGridView.HitTestInfo htinfo =
dgUsers.HitTest(e.X, e.Y);
155
156                 DataGridViewRow row = null;
157
158                 if (htinfo.RowIndex >= 0 && htinfo.ColumnIndex
>= 0)
159                     row = dgUsers.Rows[htinfo.RowIndex];
160
161                 //Código para quando se seleciona ou não uma
linha da datagrid
162                 LinhaSelecionada(row);
163             }
164             private void btnAdd_Click(object sender, EventArgs
e)
165             {
166                 //Verificar os campos de introdução obrigatória
167                 if (txtUsername.ctl_TBox.Text.Trim() ==
string.Empty ||
168                     cmbUsertype.ctl_CBox.SelectedIndex == 0)
169                 {
170                     MessageBox.Show("Os campos são
obrigatórios.", "Campos obrigatórios", MessageBoxButtons.OK,
MessageBoxIcon.Exclamation);
171                     return;
172                 }
```

```
173
174             //Liga à base de dados com um TableAdapter
175             DataSetsTableAdapters.tblLoginsTableAdapter
ta_logins = new DataSetsTableAdapters.tblLoginsTableAdapter();
176             ta_logins.Connection.ConnectionString =
functions.Get_DBConnection();
177             //Carregar a tabela
178             DataSets.tblLoginsDataTable dt_logins =
ta_logins.GetData_tblLogins(null, txtUsername.Text, null);
179
180             //Se encontrou algum registo
181             if (dt_logins.Rows.Count > 0)
182             {
183                 MessageBox.Show("O utilizador a adicionar
já existe na base de dados.", "User já existe",
MessageBoxButtons.OK, MessageBoxIcon.Exclamation);
184                 return;
185             }
186
187             //Criar um novo registo
188             DataSets.tblLoginsRow row =
dt_logins.NewtblLoginsRow();
189
190             //Atribuir os valores aos campos do registo
191             row.login = txtUsername.ctl_TBox.Text;
192
193             //Obter o ID do Tipo de Utilizador
194             int usertype_id = ((KeyValuePair<string,
int>)cmbUsertype.ctl_CBox.SelectedItem).Value;
195             row.type = usertype_id;
196
197             //Adicionar o registo à tabela
198             dt_logins.Rows.Add(row);
199
200             //Atualizar a base de dados
201             int ret = ta_logins.Update(dt_logins);
202             if (ret > 0)
203             {
204                 //Carregar os utilizadores
205                 FillUsers();
206             }
207         }
208         private void btnChange_Click(object sender,
EventArgs e)
209         {
```

```
210                 //Liga à base de dados com um TableAdapter
211                 DataSetsTableAdapters.tblLoginsTableAdapter
ta_logins = new DataSetsTableAdapters.tblLoginsTableAdapter();
212                 ta_logins.Connection.ConnectionString =
functions.Get_DBConnection();
213                 //Carregar a tabela
214                 DataSets.tblLoginsDataTable dt_logins =
ta_logins.GetData_tblLogins(Convert.ToInt32(txtID.ctl_TBox.Text)
, null, null);
215
216                 //Se o utilizador não existe
217                 if (dt_logins.Rows.Count == 0)
218                 {
219                     MessageBox.Show("O utilizador a alterar não
existe na base de dados.", "User não existe",
MessageBoxButtons.OK, MessageBoxIcon.Exclamation);
220                     return;
221                 }
222
223                 //Obter o registo
224                 DataSets.tblLoginsRow row =
((DataSets.tblLoginsRow)dt_logins.Rows[0]);
225
226                 if (row != null)
227                 {
228                     //Atribuir os valores aos campos do registo
229                     row.login = txtUsername.ctl_TBox.Text;
230
231                     //Obter o ID do Tipo de Utilizador
232                     int usertype_id = ((KeyValuePair<string,
int>)cmbUsertype.ctl_CBox.SelectedItem).Value;
233                     row.type = usertype_id;
234
235                     //Atualizar a base de dados
236                     int ret = ta_logins.Update(row);
237                     if (ret > 0)
238                     {
239                         //Carregar os utilizadores
240                         FillUsers();
241                     }
242                 }
243             }
244             private void btnRemove_Click(object sender,
EventArgs e)
245             {
```

```
246                 //Se é o último utilizador a tentar remover,
não deixar
247                 if (dgUsers.Rows.Count <= 1)
248                 {
249                     MessageBox.Show("Não é possível remover
todos os utilizadores.\r\nTem de existir pelo menos um...", "Não
remover todos", MessageBoxButtons.OK,
MessageBoxIcon.Exclamation);
250                     return;
251                 }
252
253                 //Liga à base de dados com um TableAdapter
254                 DataSetsTableAdapters.tblLoginsTableAdapter
ta_logins = new DataSetsTableAdapters.tblLoginsTableAdapter();
255                 ta_logins.Connection.ConnectionString =
functions.Get_DBConnection();
256                 //Carregar a tabela
257                 DataSets.tblLoginsDataTable dt_logins =
ta_logins.GetData_tblLogins(Convert.ToInt32(txtID.ctl_TBox.Text)
, null, null);
258
259                 //Se o utilizador não existe
260                 if (dt_logins.Rows.Count == 0)
261                 {
262                     MessageBox.Show("O utilizador a remover não
existe na base de dados.", "User não existe",
MessageBoxButtons.OK, MessageBoxIcon.Exclamation);
263                     return;
264                 }
265
266                 //Obter o registo
267                 DataSets.tblLoginsRow row =
((DataSets.tblLoginsRow)dt_logins.Rows[0]);
268
269                 if (row != null)
270                 {
271                     DialogResult resp = MessageBox.Show("Deseja
remover o utilizador selecionado?", "Remover Utilizador",
MessageBoxButtons.YesNo, MessageBoxIcon.Question,
MessageBoxDefaultButton.Button2);
272
273                     if (resp == DialogResult.Yes)
274                     {
275                         //Eliminar o registo
276                         row.Delete();
```

```
277
278                     //Atualizar a base de dados
279                     int ret = ta_logins.Update(row);
280                     if (ret > 0)
281                     {
282                         //Carregar os utilizadores
283                         FillUsers();
284                     }
285                 }
286             }
287         }
288     }
289
290     // Class de tblLogins
291     public class User
292     {
293         private int _ID;
294         private string _Username;
295         private int _UserType;
296         private string _UserTypeDesc;
297
298         public int ID
299         {
300             get { return _ID; }
301             set { _ID = value; }
302         }
303         public string Username
304         {
305             get { return _Username; }
306             set { _Username = value; }
307         }
308         public int UserType
309         {
310             get { return _UserType; }
311             set { _UserType = value; }
312         }
313         public string UserTypeDesc
314         {
315             get
316             {
317                 return _UserTypeDesc;
318             }
319             set
320             {
```

```
321                     _UserTypeDesc = value;
322                 }
323             }
324
325             // Construtor da Class
326             public User()
327             {
328
329             }
330
331             // Construtor da Class com inicialização de
variáveis
332             public User(int id, string username, int usertype)
333             {
334                 this.ID = id;
335                 this.Username = username;
336                 this.UserType = usertype;
337             }
338         }
339     }
```