PROJETO GLOBAL

RELATÓRIO

DIOGO ALEXANDRE FERREIRA DA SILVA

3º ANO LICENCIATURA EM INFORMÁTICA

Nº 8129 (Licenciatura)

Nº A044 (Pós-Graduação)

TURMA B - LABORAL

LISBOA 2017/2018

TRABALHO REALIZADO COM A ORIENTAÇÃO DE:

PROFESSOR PEDRO BRANDÃO

PROFESSORA MARIA INÊS

16 DE OUTUBRO DE 2018

Resumo

Este projeto tem como objetivo contruir uma rede local através da tecnologia de virtualização *Hyper-v* da Microsoft, que vai alojar numa das máquinas uma aplicação em *ADO.NET* para que os utilizadores dentro da rede possam utilizar. Foi utilizado uma arquitetura de *Nested Virtualization* que significa criar uma virtualização em cima de outro ambiente já virtualizado. O ambiente virtualizado que atua como base neste laboratório é constituído por uma máquina *Windows Server 2012 R2* com o *Hyper-v* que hospeda 4 máquinas virtuais. A primeira é o controlador de domínio que vai distribuir Ips fixos através do *Active Directory* às máquinas do domínio, a segunda máquina é um servidor de *Routing* que fornece *Internet* à rede, a terceira é uma máquina cliente com o *Windows 8.*1 que tem a aplicação desenvolvida em *ADO.NET* e por último temos um servidor *SQL* que vai armazenar todos os dados gravados pela aplicação.

Palavras-chave: *Virtualização, Núvem, Cloud Computing, ADO.NET, SQL, Hyper-V, Windows Server 2012 R2.*

Abstract

The purpose of this project is to build a local network through Hyper-v virtualization from Microsoft, which will host an application in ADO.NET in one machine for users within the network to use. Nested Virtualization was the architecture used, which means, creating a virtualized environment over another one. The virtualized environment that acts as a base in this lab consists in Windows Server 2012 R2 machine with Hyper-v that hosts four virtual machines. The first machine is the domain controller that will distribute fixed Ips through Active Directory to the machines of domain. The second is a Routing Server that provides Internet to the network. The third is a client machine with Windows 8.1 that has the application developed in ADO.NET. The last machine is a SQL Server that will store all the data saved by the application.

Keywords: Virtualization, Cloud, Cloud Computing, ADO.NET, SQL. Hyper-v, Windows Server 2012 R2.

Índice

INTRODUÇÃO

Este projeto está inserido no âmbito da pós-graduação em Virtualização e Cloud Computing. Tem como objetivo mostrar e compreender as principais tecnologias da Microsoft, criando uma aplicação em *Windows Forms* para gestão de utilizadores inserida numa rede de máquinas virtuais, em que cada uma tem diferente funcionalidade. O projeto está dividido em três capítulos. O primeiro começa com um estudo bibliográfico de conceitos teóricos sobre virtualização e de cloud computing como também tecnologias da *Microsoft* que foram utilizadas na pós-graduação e na licenciatura. O segundo capítulo é a contextualização que transmite ao leitor a relação entre este tema com os problemas da sociedade . Por fim temos o último capítulo que fala sobre o desenvolvimento prático do projeto, aqui vão ser analizados todos os passos que foram precisos para construir a aplicação nomeadamente também as máquinas virtuais que foram precisas para esta arquitetura funcionar.

Estado da Arte

VIRTUALIZAÇÃO

HISTÓRIA

O conceito de virtualização foi pela primeira vez abordado pela IBM na década de 60 em que foi desenvolvido uma solução de "time-sharing" que consistia no uso dos recursos do computador entre os utilizadores (Oracle, 2011). Mais tarde em Maio de 1999, a VMware lançou para o mercado a VMware Workstation, um produto com a capacidade de executatar vários sistemas operativos ao mesmo tempo num só único computador.

IMPACTO DA VIRTUALIZAÇÃO

Vivemos numa sociedade que devido ao aquecimento global e aos gases de efeito de estufa, o baixo consumo de energia tornou-se numa das grandes prioridades para os governos e empresas. Com o aparecimento já avançado da virtualização os administradores de IT conseguiam executar vários servidores virtuais independentes num só único servidor físico, permitindo assim optimizar a utilização dos recursos do servidor, e melhoramento do espaço físico e eficiência energética dos data centers. (Asian Journal of Scientific Research, 2012).

VIRTUALIZAÇÃO DE COMPUTADORES

Um computador virtual é uma representação lógica de um computador mas em software, separando o hardware físico do sistema operativo. A virtualização fornece mais flexibilidade operacional e aumenta a taxa de utilização do hardware físico subjacente. Embora a virtualização seja implementada principalmente no software, muitos microprocessadores modernos agora incluem caracteristicas no hardware explicificamente para melhorar a eficiência no processo de virtualização. Num computador tradicional, na instância do sistema operativo, suporta um ou mais

programas aplicacionais enquanto que num ambiente virtualizado, um computador físico executa um software que abstrai os recursos físicos do computador para que seja partilhado entre máquinas virtuais. Cada uma pode executar um sistema operativo diferente no mesmo computador físico. Caso que aconteça algum problema numa das máquinas virtuais não irá afetar as outras pois elas são independentes entre si. (IBM, 2007)

VIRTUALIZAÇÃO DE SERVIDORES

A virtualização de servidores permite que um servidor físico seja dividido para executar vários servidores virtuais seguros. Isto cria uma oportunidade de consolidar os servidores físicos, ajudando assim reduzir a aquisicão de hardware e os custos de administração eliminando a "expansão da infraestrutura" ao nível do servidor. Se os recursos de um servidor físico estiver na sua máxima utilização devido ao aumento do uso dos servidor virtual, movê-lo para outro servidor físico com mais recursos é tão simples como copiar um ficheiro de um computador. Essa facilidade de replicação de servidores virtuais também significa que é fácil de manter snapshots dos mesmos como também back-ups e restaurações complexas do sistema em caso que haja falhas de hardware no servidor físico. Isto aumenta os resultados da resiliência no uso de melhor eficiência dos recursos físicos do servidor, baixando os custo de operação, reduzindo a energia consumida e uma maior confiabilidade geral. (IBM, 2007)

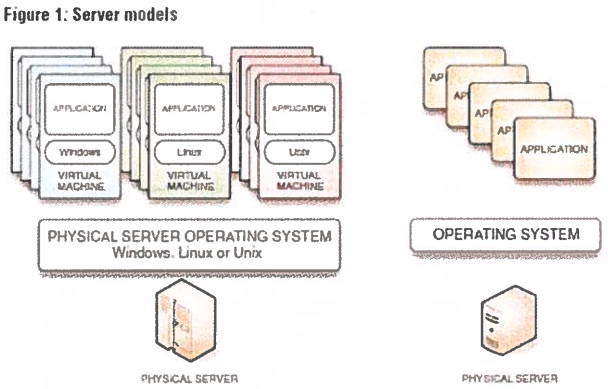


Figura 1 – Servidor Virtual e Servidor Tradicional (IBM, 2007).

TIPOS DE VIRTUALIZAÇÃO

Em virtualização, o hypervisor (conhecido também como virtual machine monitor) é um programa de baixo nível que permite múltiplos sistemas operativos sejam executados num único computador. Os Hypervisor utilizam uma camada de código em software ou em firmware para distribuir os recursos em tempo real. Existe dois tipos de hypervisors, o Tipo 1 e o Tipo 2. Os de Tipo 1, os hypervisors são executados diretamente no sistema de hardware. Eles são conhecidos como "embebidos". De acordo com a IBM, hypervisors Tipo 1 têm um melhor desempenho, eficácia e segurança do que o Tipo 2 hypervisor. Exemplo, Hyper-V da Microsoft e VMware ESX/ESXi. Os de Tipo 2 são hypervisors que são executados no sistema operativo. Exemplos, VMware Workstation e VirtualBox. (Techtarget, 2010).

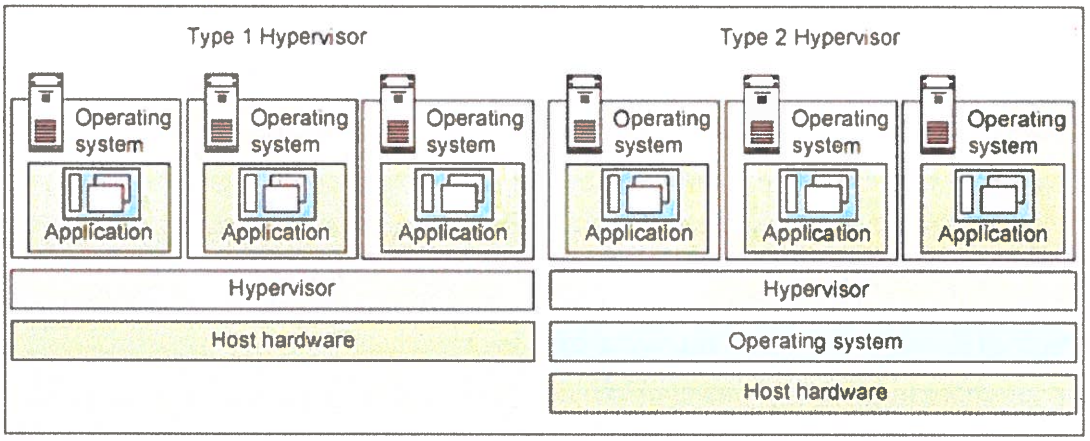


Figura 2 – Virtualização Tipo 1 e Tipo 2 (IBM, 2011).

STRUCTURED QUERY LANGUAGE (SQL)

Structured Query Language é uma linguagem de consulta estruturada para relacionar base dados e realizar várias operações de dados entre eles. Foi inicialmente criado em 1970, SQL é regularmente usado pelos administradores de base de dados, por programadores que utilizam scripts de dados nos seus códigos como também analistas de dados. Os usos do SQL incluem, a modificação das tabelas da base de dados, adicionar, apagar e atualizar as linhas dos dados, buscar a informação dos dados em subconjuntos. As queries (consultas) e outras operações SQL são executadas em forma de comando como por exemplo, select, add, insert, update, delete, create, alter e truncate. Os comandos do SQL estão divididos em diferentes tipos, entre eles data manipulation language (DML) e data definition language (DDL), controle de transações e medidas de segurança. O vocabulário DML é usado para buscar e manipular dados, enquanto que as expressões DDL são usadas para definir e modificar estruturas da base de dados. O controle de transações ajuda no processo de transação, assegurando que as mesmas estejam concluídas ou caso que algum erro aconteça faça um rollback. As medidas de segurança são usadas para controlar o acesso à base de dados mas também para criar as funções e permissões dos utilizadores (TechTarget, 2016).

SQL SERVER 2014

O Sql Server 2014, incorpora a nova era da Microsoft Cloud OS, que fornece às organizações e clientes uma plataforma consistente para a infraestrutura, aplicações, serviços de alojamento de data centers, e Microsoft public cloud. Os benefícios que os clientes irão experienciar com esta plataforma inclui o desenvolvimento comum, gestão, dados, identidade e virtualização, não importa onde a aplicação está sendo executada. SQL Server 2014 também oferece às organizações a oportunidade de eficientemente proteger, desbloquear, e aumentar os seus dados sobre os computadores, dispositivos móveis, data centers e sobre public, private ou hybrid cloud (Microsoft, 2014).

SQL SERVER 2014 EDITIONS

SQL Server 2014 está disponível em três edições. Todas elas têm um alinhamento mais apertado do que os seus predecessores e foram projetadas para atender às necessidade de qualquer cliente. Cada edição vem com uma versão de 32 bits e 64 bits (Microsoft). A edição Enterprise do SQL Server 2014 é o SKU mais alto e é considerada a oferta premium. É um produto projetado para satisfazer as mais altas demandas de centros de dados em larga escala e soluções de data warehouse, fornecendo desempenho e disponibilidade de missão crítica para aplicativos de nível 1 e a capacidade de implementar private-cloud, ambientes virtualizados e soluções de business-intelligence. Na edição Standard é considerada uma plataforma de gerenciamento de dados adaptada a base de dados departamentais e aplicativos de business-intelligence limitados que são tipicamente apropriados para organizações pequenas ou para departamentos. Não inclui todas as características de Enterprise e Businnes-intelligence embora continue oferecendo a melhor capacidade de gerenciamento e facilidade de uso. Em relação às outras edições esta suporta até 16 núcleos. Na edição Business Intelligence é oferecido às organizações o conjunto completo de poderosos recursos de BI, como relatórios e análises, Power View e PowerPivot. É adapatado às organizações que precisam de Business Intelligence corporativa e capacidades de autoatendimento, mas que não exigem o desempenho e a escalabilidade de processamento transacional on-line (OLTP) (Microsoft, 2014).

ADO.NET

O ADO.NET fornece acesso consistente a fontes de dados como o SQL Server e o XML, e a fontes de dados expostas através do OLE DB e do ODBC. Os aplicativos do consumidor de compartilhamento de dados podem usar o ADO.NET para se conectar a essas fontes de dados, e para recuperar, manipular e atualizar os dados nelas contidos.O ADO.NET separa o acesso a dados da manipulção de dados em componentes discretos que podem ser usados separamente e inclui os provedores de dados do .NET Framework

para se conectar a um banco de dados, executar comandos e recuperar resultados. Esses resultados são processados diretamente, colocados em objeto DataSet do ADO.NET para serem expostos para o usuário ad hoc, combinados com dados de várias fontes ou passados entre as camadas. O objeto DataSet também poe ser usado independentemente de um provedor de dados .NET Framework para gerenciar o local dos dados para o aplicativo ou originado no XML. As classes do ADO.NET estão no System.Data.dll e são integradas às classes XML encontradas no System.Xml.dll. Para o código de exemplo que se conecta a um banco de dados recupera dados dele e, em seguida, exibe dados em uma janela de console. O ADO.NET fornece a funcionalidade para os desenvolvedores que gravam um código gerenciado semelhante à funcionalidade fornecida aos desenvolvedores COM nativos pelos objetos ActiveX Data Objects (ADO). É recomendável que se use o ADO.NET,e não o ADO, para acessar dados nos aplicativos .NET. O ADO.NET fornece o método mais direto de acesso a dados no .NET Framework. Para uma abstração de alto nível que permite que os aplicativos funcionem em um modelo conceitual em vez de do modelo de armazenamento subjacente (Microsoft).

CLOUD COMPUTING

CONCEITO

Cloud Computing é um modelo ubíquo, conveniente e sob demanda, que dá acesso a uma rede que partilha os recursos configuráveis de computação (redes, servidores, aramazenamento, aplicações e serviços) que podem ser rapidamente provisionados e libertados com o mínimo esforço de gestão ou interação do provedor de serviços. Este modelo da cloud é composto por cinco características, três modelos de serviço e quatro modelos de implementação (Peter Mell, Tim Grance, 2011).

CARACTERÍSTICAS

Serviço *Self-Service* – O consumidor pode, unilateralmente, requerer ou dispensar capacidades de computação, tais como o tempo do servidor, a capacidade de armazenamento, ou outros, conforme necessário e de forma automática. Tudo, sem necessidade de interação humana com o forncedor de cada serviço. Acesso à rede em banda larga – Todas as funcionalidades estão disponíveis através da rede e são acessíveis por meio de mecanismos *standard*, que promovem o uso de plataformas-cliente heterogéneas (telefones móveis, laptops, PDAs, etc). Pool de recursos – Os recursos de computação de cada fornecedor são concebidos para servir vários clientes, num modelo *multi-tenant* , com diferentes recursos físicos e virtuais, distribuídos e alocados dinamicamente. Elasticidade – Os recursos podem ser rapidamente alocados e, em alguns casos, de forma automática, para aumentar as capacidades disponíveis ou para as libertar quanto já não são necessárias. Parao cliente, os recursos de alocação têm inúmeras possibilidades, podendo ser adquiridas em qualquer quantidade e a qualquer momento. Mensurável – Os sistemas em cloud devem controlar e optimizar a utilização dos recursos de forma automática, efectuando a medição da utilização, de forma adequada ao tipo de serviço, como por exemplo, armazenamento utilizado, processamento efectuado, largura de banda utilizada ou contas de utilizadores ativas. O uso dos recursos deve poder ser monitorizado e controlado de forma transparente, tanto para o fornecedor, como para o consumidor do serviço utilizado (DXC.technology).

MODELOS DE SERVIÇO

O software como serviço (SaaS) é um método para fornecer aplicativos de software pela Internet, sob demanda e, normalmente, em uma base de assinaturas. Com o SaaS, os provedores de nuvem hospedam e gerenciam o aplicativo de software e a infraestrutura subjacente e fazem manutenções, como atualizações de software e aplicação de patch de segurança. Os usuários conectam o aplicativo pela Internet, normalmente com um navegador da Web em seu telefone, tablet ou PC. A plataforma como serviço (PaaS) refere-se aos serviços de computação em nuvem que fornecem um ambiente sob demanda para desenvolvimento, teste, fornecimento e gerenciamento de

aplicativos de software. O PaaS foi criado para facilitar aos desenvolvedores criarem aplicativos móveis ou Web rapidamente, sem se preocupar com a configuração ou o gerenciamento de infraestrutura subjacente de servidores, armazenamento, rede e bancos de dados necessários para desenvolvimento. A infraestrutura como serviço (IaaS) o cliente aluga infraestrutura de TI, servidores e VMs (máquinas virtuais), armazenamento, redes e sistemas operacionais, de um provedor de nuvem em uma base pré-paga (Microsoft).

TIPOS DE IMPLEMENTAÇÃO

A public cloud são de propriedade de um provedor de serviços de nuvem de terceiros e operadas por ele, que por sua vez fornece recursos de computação, como servidores e armazenamento pela Internet. O Microsoft Azure é um exemplo de public cloud. Com este tipo de cloud, todo o hardware, software e outras infraestruturas de suporte são de propriedade e gerenciadas pelo provedor da cloud. Estes serviços podem ser acedidos usando um navegador da Web. A private cloud refere-se aos recursos de computação em cloud usados exclusivamente por uma única empresa ou organização. Pode estar localizada fisicamente no datacenter local da empresa. Algumas empresas também pagam provedores de serviço de terceiros para hospedar a sua private cloud. Os serviços e infraestrutura são mantidos em uma rede privada (Microsoft). A hybrid cloud é a combinação de private e public cloud ligadas por uma tecnologia que permite que dados e aplicativos sejam compartilhados entre elas. Ao permitir que dados e aplicativos sejam movidos entre clouds, a hybrid cloud dá aos negócios mais flexibilidade e mais opções de implantação (Microsoft).

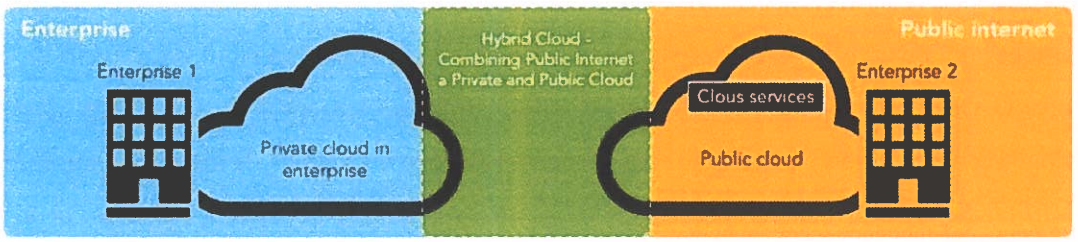


Figura 3 – Public, Private e Hybrid Cloud

Fonte:

https://www.dialogic.com/~/media/products/docs/whitepapers/12023-cloud-computing-wp.pdf

BENEFÍCIOS

Poupança de custos – As empresas podem reduzir as suas despesas de capital e usar as suas despesas operacionais para aumentar as capacidades de computação. Escalabilidade/Flexibilidade – As empresas podem começar com uma pequena implementação e crescer para uma grande implementação com bastante rapidez, e reduzir se for necessário. Além disso, a flexibilidade da computação em cloud permite que as empresas usem recursos extras nos horários com maior utilização, permitindo satisfazer o pedido dos consumidores. Confiabilidade – Os serviços que utilizam vários sites redundantes podem suportar a continuidade do negócio e recuperação de sistemas. Manutenção – Os fornecedores de serviços em cloud fazem a manutenção do sistema e o acesso é através de APIs que não requerem instalações nos computadores, reduzindo mais ainda os requesitos de manutenção. Acessível – Os trabalhadores móveis aumentaram a produtividade devido a sistemas acessíveis numa infra-estrutura disponível em qualquer lugar (Dialogic White Paper, 2017).

SYSTEM CENTER VIRTUAL MACHINE MANAGER

Virtual Machine Manager faz parte do System Center Virtual Machine Manager, que é usado para configurar, administrar e transformar datacenters tradicionais, e ajudar a fornecer uma experiência de gerenciamento em qualquer lugar, serviços e a núvem do Azure. O Datacenter ajuda a configurar e administrar os elementos do datacenter que abrange servidores de virtualização, componentes de rede e recursos de armazenamento. O Virtual Machine Manager prevê e administra os recursos necessários para criar e implantar máquinas e serviços virtuais para núvens privadas. Na virtualização o Virtual Machine Manager pode adicionar, provisionar e gerir hosts e clusters de virtualização do VMware e Hyper-V. Na rede o Virtual Machine Manager fornece virtualização de rede,

incluindo suporte para criar e gerenciar redes virtuais e gateways de rede. A virtualização de rede permite que vários inquilinos tenham redes isoladas e os seus próprios intervalos de endereços IP para maior privacidade e segurança. Usando gateways, as máquinas virtuais em redes virtuais podem se conetar a redes físicas no mesmo site ou em locais diferentes. No armazenamento o Virtual Machine Manager pode classificar, provisionar, alocar e atribuir armazenamento local e remoto. Os recursos da biblioteca do Virtual Machine Manager mantém uma biblioteca de recursos baseados e não baseados em ficheiro que são usados para criar e implantar máquinas virtuais. Os recursos baseados em ficheiros incluem discos rígidos virtuais, imagens ISO e scripts. Os recursos não baseados em ficheiros incluem modelos e perfis que são usados para seguir um padrão de criação de máquinas virtuais. Os recursos da biblioteca são acedidos através do compartilhamento de bibliotecas (Microsoft).

II. CONTEXTUALIZAÇÃO

Nos dias de hoje, os dados são o novo ouro do seculo XXI. Para responder a este crescimento exponecial de todos os anos, as empresas precisavam de uma maneira de armazenar estas grandes quantidades de dados (Big Data) para depois mais tarde conseguirem tratar e utilizálos, nomeadamente para fazer estatistica de consumos futuros como também preferências de cada público alvo. Foi então que surgiu a Cloud, uma tecnologia revolucionária para dar solução a este tipo de situação. Com a Cloud as empresas deixaram-se de preocupar em adquirir o espaço físico para focarem-se no core do seu negócio e começaram a utilizar ambientes virtualizados de empresas especializadas como por exemplo Microsoft que contém ínumeras ferramentas para responder às necessidades do consumidor/empresas.

Desenvolvimento

Neste laboratório foram necessárias 5 máquinas virtuais, 4 delas (*DC*, *Routing*, *Storage* e *Hyper-V*) têm o sistema operativo *Windows Server 2012 R2*, a restante máquina virtual (WIN 8.1) tem o sistema operativo *Windows 8.1*, pois vai ser a nossa máquina cliente que vai ter a aplicação. Em relação à conetividade entre máquinas, foram criados dois *Virtual Switches*, um *Private*, para todas as máquinas do domínio conseguirem comunicar entre si e um *External* para fazer conectividade com a máquina física.

A primeira máquina virtual principal que tem como nome *Hyper-V* vai ser o *Hypervisor* unicamente que vai hospedar as restantes 4 máquinas virtuais (*DC, Routing, Storage e WIN 8.1*).

A segunda máquina, *DC*, é o controlador de domínio, para ter esta função foi preciso primeiro definir a rede, para tal foi escolhido o IP 10.1.1.0 e atribui-se o ip fixo 10.1.1.2 a esta máquina. A seguir instalou-se o *Active Directory* e criou-se o domínio 'projetoglobal.local'. É nesta máquina também que vão ser criados os utilizadores que podem aceder à aplicação.

Na terceira máquina, *Routing*, vai ter como função fornecer *Internet* às máquinas dentro do domínio. Ao contrário das outras 3 máquinas (*DC, Storage, WIN 8.1*) que só têm uma placa de rede através do *Private Virtual Switch*, esta tem duas, uma com o *Private Virtual Switch*, que funciona como *gateway* da rede local com o IP 10.1.1.15 , e uma com o *External Virtual Switch* que através do *NAT* vai conseguir fornecer *Internet* às máquinas do domínio.

Na quarta máquina temos o servidor de base de dados, *Storage*, com o IP 10.1.1.13, que vai armazenar todos os dados criados pela aplicação, para isso instalou-se o *SQL Server 2012*. Para o bom funcionamento entre a aplicação e este servidor, tem-se verificar sempre se os serviços *SQL* estão a ser executados.

Por último temos a máquina cliente, *WIN 8.1*, com o IP 10.1.1.8, onde vai ter a aplicação que foi criada.

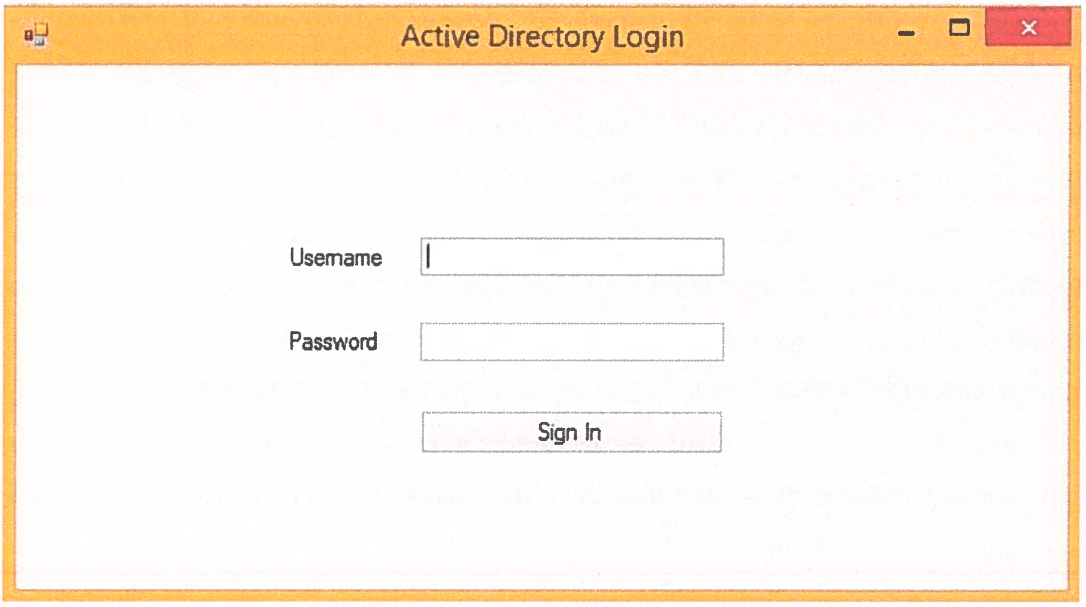


Figura 4. Interface da aplicação. Fonte: Do Autor

Assim que se inicia a aplicação este é o *layout* que vai ser apresentado (fig. 4). É um sistema de *login* composto por duas textbox que vão fazer autenticação do *input* do utilizador. As duas *textbox* são de preenchimento obrigatório. Esta autenticação só é permitida a utilizadores que tenham sido adicionados no *Active Directory*.

```
private void btnLogin_Click(object sender, EventArgs e)
{
    //verificação dos campos
    if(txtUsername.Text != "" && txtPassword.Text != "")
    {
        //Ligação ao dominio
        string path = @"LDAP://DC.projetoglobal.local";
        string dominio = @"projetoglobal.local";
        string username = txtUsername.Text.Trim();
        string password = txtPassword.Text.Trim();
        string domUser = dominio + @"\" + username;
        //varificação autenticação dos dados inseridos
        bool loginSuccess = Authenticate(path, domUser, password);
        if (loginSuccess)
        {
            MessageBox.Show("Login Successful!");
            Form1 fAdmin = new Form1(txtUsername.Text);
            fAdmin.Show();
            this.Hide();
        }
        else
        {
            MessageBox.Show("Access Denied");
```

Figura 5. Criação das variáveis para verificação. Fonte: Do Autor

Como mostra na figura 5 foram criadas 5 *strings*, a *string path* contém o caminho do *LDAP*, a *string* dominio é o nome do dominio que foi criado, as *strings username* e *password* vão receber o *input* das *textbox username* e *password* respetivamente. A *string domUser* vai concatenar a *string* dominio com a de *username*, que mais tarde vai ser usada na função *Authenticate* (ver fig. 6). Esta recebe três parâmetros, que vão ficar guardadas no objeto *de* que depois vai ser utilizado com o *DirectorySearcher*, se encontrar o utilizador retorna *true* (1 booleano) se não retorna *false* (0 booleano), ver figura 6. O resultado desta função vai ser utilizado para a variável boleana *loginSuccess*, se a autenticação for válida vai ser mostrada a mensagem *"Login Successful"* e redirecionar o utilizador para a interface de gestão (*Form1*), caso contrário irá ser exibida a mensagem *"Access Denied"*.

```
//Função da autenticação ao dominio
private bool Authenticate(String path, String username, String password)
{
    DirectoryEntry de = new DirectoryEntry(path, username, password, AuthenticationTypes.Secure);
    try
    {
        DirectorySearcher ds = new DirectorySearcher(de);
        ds.FindOne();
        return true;
    }
    catch
    {
        return false;
    }
}
```

Figura 6. Autenticação dos dados inseridos. Fonte: Do Autor

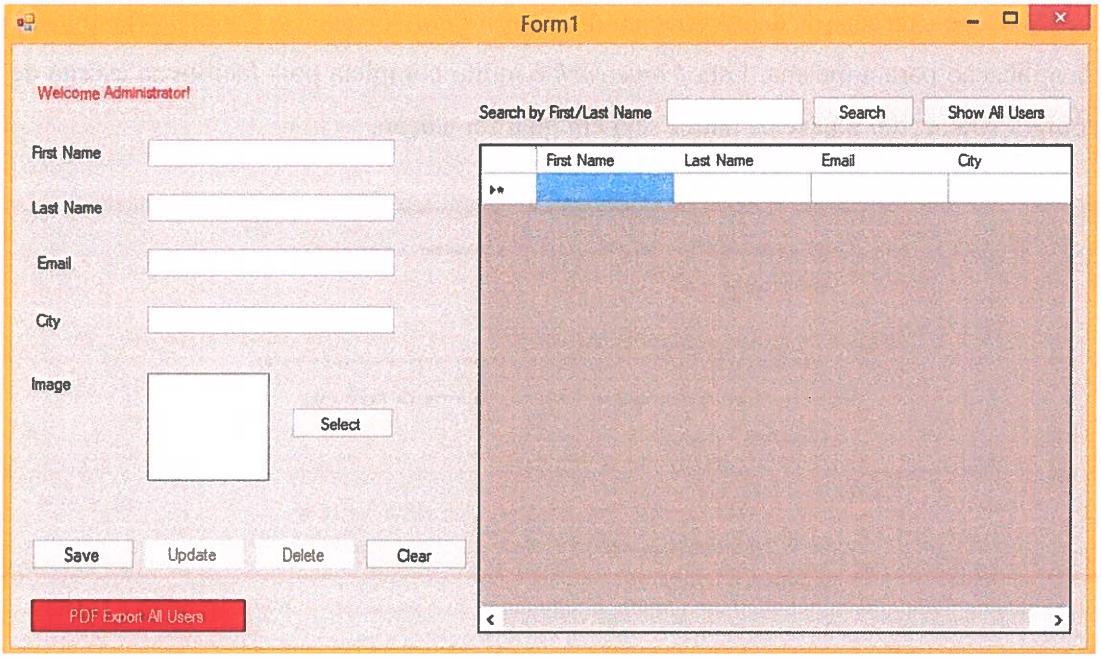


Figura 7. Interface gestão de utilizadores. Fonte: Do Autor

Como se pode verificar na figura 7 este é a interface de gestão de utilizadores. O interface está dividido em duas partes, na parte esquerda temos os campos que podem ser utilizados para as ações gravar, atualizar e eliminar, na parte direita encontra-se a pesquisa de dados em que os resultados vão aparecer numa grelha.

Para gravar os dados teve-se que criar primeiro uma base de dados na máquina *Storage*. Deu-se como nome *myDataBase* e contém apenas uma tabela com 6 colunas, o ID do tipo int que vai ser a chave primária e única, o *First Name, Last Name, Email e City* são do tipo *varchar* com máximo de 50 caracteres e a *Image* é do tipo *image* (ver fig. 8).

Column Name	Data Type	Allow Nulls
Id	int	☐
firstName	varchar(50)	☐
lastName	varchar(50)	☐
email	varchar(50)	☐
city	varchar(50)	☐
image	image	☐

Figura 8. Tabela users da base de dados myDataBase. Fonte: Do Autor

Depois da base de dados criada utilizou-se a *Entity Framework* para fazer a ligação da aplicação para a mesma. Esta *framework* é muito completa pois facilita na escrita de código para aceder à base de dados seja em qual for a ação.

```
//Aceitar no campo First Name só letras
private void txtFirstName_KeyPress(object sender, KeyPressEventArgs e)
{
    if (!char.IsControl(e.KeyChar) && !char.IsLetter(e.KeyChar))
    {
        e.Handled = true;
    }
}
//Aceitar no campo Last Name só letras
private void txtLastName_KeyPress(object sender, KeyPressEventArgs e)
{
    if (!char.IsControl(e.KeyChar) && !char.IsLetter(e.KeyChar))
    {
        e.Handled = true;
    }
}
//Aceitar no campo City só letras
private void txtCity_KeyPress(object sender, KeyPressEventArgs e)
{
    if (!char.IsControl(e.KeyChar) && !char.IsLetter(e.KeyChar))
    {
        e.Handled = true;
    }
}
```

Figura 9. Restrições para os campos *First Name, Last Name e City*. Fonte: Do Autor

Antes de qualquer gravação para a base de dados existe um conjunto de requesitos que impede o mesmo. Nas *textbox First Name, Last Name* e *City* não são permitidos nem números nem caracteres especiais (ver fig 9). Mesmo que o utilizador tente escrever com uma destas duas não irá aparecer nada. Estes métodos (*txtFirstName_KeyPress, txtLastName_KeyPress e txtCity_KeyPress*) estão a verificar se a tecla pressionada pelo utilizador é uma letra e caso se for, é escrita devolta para a *textbox*.

```
//Verificação se o email é válido ou não
private bool IsValidEmail(string strIn)
{
    bool invalid = false;
    if (String.IsNullOrEmpty(strIn))
        return false;

    strIn = Regex.Replace(strIn, @"(@)(.+)$", this.DomainMapper);
    if (invalid)
        return false;

    return Regex.IsMatch(strIn,
      @"^(?(")(""[^""]+?""@)|((([0-9a-z]((\.(?!\.))|[-!#\$%&'\*\+/=\?\^`\{\}\|~\w])*)(?<=[0-9a-z])@))" +
      @"(?(\[)(\[(\d{1,3}\.){3}\d{1,3}\])|(([0-9a-z][-\w]*[0-9a-z]*\.)+[a-z0-9]{2,17}))$",
      RegexOptions.IgnoreCase);

}
```

Figura 10. Verificação de email. Fonte: Do Autor

Na figura 10 mostra a verificação no campo de *email*. Através do método *IsValidEmail*, vai receber uma variável do tipo *string* que depois através do método *Regex* vai verificar se o *email* está num formato válido ou seja se tem pelo menos o caractere @. É de anotar que este método não verifica se o email introduzido é real.

```
//Função para converter uma imagem em Byte
private byte[] ConvertFiltoByte(string sPath)
{
    byte[] data = null;
    FileInfo fInfo = new FileInfo(sPath);
    long numBytes = fInfo.Length;
    FileStream fStream = new FileStream(sPath, FileMode.Open, FileAccess.Read);
    BinaryReader br = new BinaryReader(fStream);
    data = br.ReadBytes((int)numBytes);
    return data;
}
```

Figura 11. Conversão de imagem para *byte*. Fonte: Do Autor

No campo *image* podemos escolher a imagem que queremos que seja gravada, mas para que isso aconteça, primeiro precisamos de converter a imagem em *bytes.* Como mostra na figura 11, o método *ConvertFiltoByte* faz esse processo. Este é do tipo *byte* e recebe como argumento uma *string* que por sua vez é o caminho onde a imagem está guardada no computador do utilizador. É criada uma variável *data* do tipo *byte,* a seguir cria-se um objeto *fInfo* da classe *FileInfo* que nos vai dar informação à cerca do ficheiro selecionado. De seguida calcula-se o número de *bytes* que este ficheiro tem e através da classe *FileStream* que tem como objetivo manipular ficheiros (abrir, ler, escrever, e fechar) cria-se um objeto *fStream.* Com a classe *BinaryReader* vai ler o número *bytes* que foram especificados pela variável *numBytes* e guardar tudo numa variável *data.*

```
//Função para converter Byte em imagem
private System.Drawing.Image ConvertBytetoImage(byte[] photo)
{
    System.Drawing.Image newImage;
    using (MemoryStream ms = new MemoryStream(photo, 0, photo.Length))
    {
        ms.Write(photo, 0, photo.Length);
        newImage = System.Drawing.Image.FromStream(ms, true);
    }
    return newImage;
}
```

Figura 12. Conversão *byte* em imagem. Fonte: Do Autor

Quando a imagem é guardada na base de dados e queremos voltar a vizualizar é necessário converter os *bytes* da imagem. Com o método *ConvertBytetoImage* esta transformação resliza-se pelos seguintes passos. Primeiro é criado uma variável *newImage* do tipo *System.Drawing.Image*. Esta classe cria uma imagem por um conjunto de dados especifico de uma *stream*, pelo método *FromStream*. Esta *stream* por sua vez é criada através da classe *MemoryStream* pelo método *Write* que vai escrever um bloco de *bytes* que estão a ser lidos.

```
private void btnSave(object sender, EventArgs e)
{
    try
    {

        var teste = txtEmail.Text;
        if (txtFirstName.Text != "" && txtLastName.Text != "" && txtEmail.Text != "" && txtCity.Text != "" && imageView != null)
        {
            if (IsValidEmail(teste) == true)
            {
                model.firstName = txtFirstName.Text.Trim();
                model.lastName = txtLastName.Text.Trim();
                model.email = teste;
                model.city = txtCity.Text.Trim();
                model.image = ConvertFiltoByte(this.imageView.ImageLocation);
                using (myDataBaseEntities db = new myDataBaseEntities())
                {
                    db.users.Add(model);
                    db.SaveChanges();
                }
                clear();
                DataGridView();
                MessageBox.Show("Saved");
            }
            else
            {
                MessageBox.Show("Please enter a valid email address!");
            }
        }
        else
        {
            MessageBox.Show("Please fill all the fields");
        }


    }
    catch (Exception ex)
    {
        throw new ApplicationException("Something went wrong...", ex);
```

Figura 12. Inserção de dados. Fonte: Do Autor

Como mostra a figura 12 todos os campos têm de ser preenchidos obrigatoriamente para serem gravados. Caso isto for verdadeiro, o email inserido vai ser analizado por outro método (*IsValidEmail*) e o resultado dessa verificação vai ditar a continuação da gravação. O objeto model é nome que se deu quando se fez a ligação da *Entity Framework* à base de dados. Esse model contêm todas as colunas da tabela users. Funcionado como um objeto basta corresponder os campos que foram preenchidos pelo utilizador à respetiva coluna da tabela. Assim que este objeto tiver todas propriedades preenchidas resta apenas gravar na base de dados, para isso é necessário criar um contexto, que tem pelo nome *db* e associar este objeto ao contexto através do metodo *Add* e de seguida gravar as alterações.

```
//Função para listar todos os utilizadores da base de dados
void DataGridView()
{
    dataGridViewUsers.AutoGenerateColumns = false;
    using (myDataBaseEntities db = new myDataBaseEntities())
    {
        dataGridViewUsers.DataSource = db.users.ToList<user>();
    }
}
```

Figura 13. Listar todos os registos da base de dados. Fonte: Do Autor

```
//Deixa algumas textbox indisponíveis e duplo clique numa linha do gridview passa os valores selecionados para as textbox
private void dataGridViewUsers_DoubleClick(object sender, EventArgs e)
{

    btnSave_field.Enabled = false;
    btnUpdate_field.Enabled = true;
    btnDelete_field.Enabled = true;
    txtFirstName.Enabled = false;
    txtLastName.Enabled = false;


    if(dataGridViewUsers.CurrentRow.Index != -1)
    {
        model.Id = Convert.ToInt32(dataGridViewUsers.CurrentRow.Cells["Id"].Value);
        using(myDataBaseEntities db = new myDataBaseEntities())
        {
            model = db.users.Where(x => x.Id == model.Id).FirstOrDefault();
            txtFirstName.Text = model.firstName;
            txtLastName.Text = model.lastName;
            txtEmail.Text = model.email;
            txtCity.Text = model.city;
            imageView.Image = ConvertBytetoImage(model.image);
        }
    }

}
```

Figura 14. Passagem dos dados da *GridView* para as *Textbox.* Fonte: Do Autor

Na figura 7, existe uma *GridView* que vai mostrar todos os resultados provenientes da base de dados quando o utilizador clicar no botão *Show All Users.* Para processo acontecer basta apenas uma query à base de dados através do método *db.users.ToList<user>* (ver figura 13). Tendo agora a listagem de todos os registos, cada linha que aparece é considerada uma *row.* em que podemos selecionar (clicando duas vezes). Após seleção os dados dessa *row* vão ser transferidos para as *Textbox* respetivas, através do objeto *model* (ver figura 14) e assim já conseguimos atualizar ou eliminar dados.

```
//Atualiza valores que foram inseridos para a base de dados
private void btnUpdate(object sender, EventArgs e)
{

    model.firstName = txtFirstName.Text;
    model.lastName = txtLastName.Text;
    model.email = txtEmail.Text;
    model.city = txtCity.Text;
    model.image = ConvertFiltoByte(this.imageView.ImageLocation);

    using (myDataBaseEntities db = new myDataBaseEntities())
    {
        db.Entry(model).State = EntityState.Modified;
        db.SaveChanges();
    }
    MessageBox.Show("Update Complete");
    DataGridView();
    clear();
    btnSave_field.Enabled = true;
    btnUpdate_field.Enabled = false;


}
```

Figura 15. Atualizar dados. Fonte: Do Autor

Atualizar os dados é semelhante à gravação de dados em que os dados que foram linkados através da *Gridview* para as *textbox*. O utilizador assim que acabar as suas alterações, ao clicar no botão update os dados que estão nas *Textbox* vão para o objeto *model* de acordo com a sua respetiva propriedade e no final através do método *db.Entry().State* os dados da *row* que está a ser editada vão ser gravados com as novas alterações (ver figura 15).

```
//Elimina o registo secionado
private void btnDelete(object sender, EventArgs e)
{
    if (MessageBox.Show("Are you sure want to delete?", "Delete", MessageBoxButtons.YesNo) == DialogResult.Yes)
    {
        using (myDataBaseEntities db = new myDataBaseEntities())
        {
            var entry = db.Entry(model);
            if (entry.State == EntityState.Detached)
            {
                db.users.Attach(model);
            }
            MessageBox.Show("User Removed");
            db.users.Remove(model);
            db.SaveChanges();
```

Figura 16. Eliminar dados. Fonte: Do Autor

Por outro lado, caso se o utilizador pretender eliminar dados depois ter selecionado a *row,* assim que clicar no botão *delete*, irá aparecer a seguinte mensagem "*Are you sure want to delete?"* com duas opções, *"Yes"* ou *"No"*(ver figura 16).

Escolhendo a opção *"Yes"* o evento *btnDelete* vai primeiro verificar qual o estado que o objeto *model* que se encontra (este objeto *model* pode ser revisto na figura 13). Caso que o seu estado tenha sido removido, ele volta a ser anexado com as propriedades todas que estão nas *Textbox.* Depois disto, através do contexto *db* e com o método *Remove*, os dados da *row* são eliminados.

```
//Botão para pesquisar por primeiro nome ou pelo último
private void btnSearch_Click(object sender, EventArgs e)
{
    dataGridViewUsers.AutoGenerateColumns = false;

    using (myDataBaseEntities db = new myDataBaseEntities())
    {
        string nomeSearch = txtSearch.Text;
        dataGridViewUsers.DataSource = db.users.Where(x => x.firstName.StartsWith(nomeSearch) || x.lastName.StartsWith(nomeSearch)).ToList();
    }

}
```

Figura 17. Pesquisar registo na base de dados. Fonte: Do Autor

Se o o utilizador eventualmente necessitar de pesquisar algum registo, para não estar a procurar diretamente na *GridView* verificando um a um, ele pode filtrar a sua pesquisa, digitando na *Textbox Search by First/Last Name* o respetivo nome. Para isso ao clicar no botão *Search* vai disparar o evento *btnSearch* (ver figura 17). Este método vai criar uma variável do tipo *string, nomeSearch,* que contém a palavra que o utilizador inseriu. Posteriormente realizou-se uma *query* à base de dados com o método *db.users.Where()*, com a cláusula *Where* podemos definir que tipo de resultados podemos filtrar e mostrar. Neste caso a palavra que o utilizador inserir a query vai procurar nas colunas, *firstname* e *lastname.*

```
//Extrair todos os dados que estão na base de dados para um PDF
private void btnPdfExport(object sender, EventArgs e)
{
    using(SaveFileDialog sfd = new SaveFileDialog() { Filter="Pdf file|*.pdf",ValidateNames = true })
    {
        if (sfd.ShowDialog() == DialogResult.OK)
        {
            iTextSharp.text.Document doc = new iTextSharp.text.Document(PageSize.A4.Rotate());
            try
            {

                PdfWriter.GetInstance(doc, new FileStream(sfd.FileName, FileMode.Create));
                doc.Open();
                PdfPTable table = new PdfPTable(6);
                PdfPCell cell = new PdfPCell(new Phrase("Users"));
                cell.Colspan = 6;
                cell.HorizontalAlignment = 1;
                table.AddCell(cell);
                table.AddCell("ID");
                table.AddCell("First Name");
                table.AddCell("Last Name");
                table.AddCell("Email");
                table.AddCell("City");
                table.AddCell("Image");
```

Figura 18. Extrair dados da GridView para PDF Parte 1/2. Fonte: Do Autor

```
    using (myDataBaseEntities db = new myDataBaseEntities())
    {
        var dataUser = db.users.Select(x => new
        {
            Id = x.Id.ToString(),
            firstName = x.firstName,
            lastName = x.lastName,
            email = x.email,
            city = x.city,
            image = x.image


        }).ToList();

        foreach (var userInfo in dataUser)
        {
            table.AddCell(userInfo.Id);
            table.AddCell(userInfo.firstName);
            table.AddCell(userInfo.lastName);
            table.AddCell(userInfo.email);
            table.AddCell(userInfo.city);
            iTextSharp.text.Image img = iTextSharp.text.Image.GetInstance(userInfo.image);
            table.AddCell(img);
        }
    }

    doc.Add(table);

    doc.Close();
    MessageBox.Show("File Created");
}
catch (Exception ex)
{

    MessageBox.Show(ex.Message, "Message", MessageBoxButtons.OK, MessageBoxIcon.Error);
}
finally
{
    doc.Close();
}
```

Figura 19. Extrair dados da GridView para PDF Parte 2/2. Fonte: Do Autor

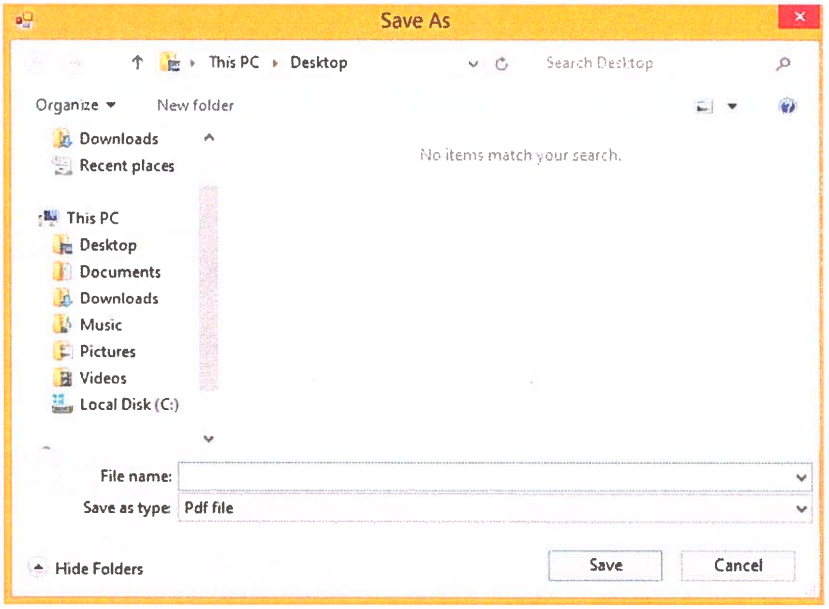


Figura 20. Localização para gravar o PDF. Fonte: Do Autor

Se o utilizador clicar no botão *PDF Export All Users,* vai extrair todos os registos que estão na base de dados para um *PDF*. Através do evento *btnPdfExport* vai ser criado um objeto *sfd* da classe *SaveFileDialog* que vai abrir uma janela (ver figura 20) que vai ser o local onde o ficheiro *PDF* vai ser guardado. A seguir através da classe *iTextSharp.text.document* vai ser criado um objeto *doc* em que vai ser definido as dimensões do *PDF*, neste caso foi A4 (ver figura 18). Com as classes *PdfPTable* e *PdfPCell* vão estipular quantas colunas vão ter este *PDF* em que cada coluna vai representar uma célula que vai ter como título cada campo da tabela da base de dados. Definindo as colunas, é necessário agora fazer uma *query* à base de dados para guardar todos os registos para uma variável, *dataUser* (ver figura 19). Para aceder a esta variável que contém todas as informações dos registos e associar a cada célula o respetivo campo, foi necessário criar um *foreach* que vai percorrer cada *index* da *dataUser* para depois ser agregado ao objeto *table* e por fim ser guardado ao objeto *doc*. Este é o aspeto do resultado final da criação do *PDF* (ver figura 21).

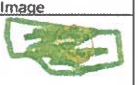
Users					
ID	First Name	Last Name	Email	City	Image
32	mario	fr	asd@mail.com	Caldas da Rainha	
34	Diogo	Silva	diogo@mail.com	Lisboa	

Figura 21. Visualização do conteúdo do *PDF*. Fonte: Do Autor

CONCLUSÃO

Em função do objetivo de estudo chegou-se à conclusão que existe uma enorme dependência da virtualização desde das micro empresas até grandes empresas. Isto deve-se ao facto de o crescimento da nossa sociedade ser demasiado exponencial e para tal tem de existir uma tecnologia para responder a essas necessidades. Assim estas empresas conseguem-se focar no seu *Core Business,* diminuindo os custos e aumentando a produtividade que consequentemente aumentará as suas receitas. Com o aparecimento também da *Cloud,* a facilidade com que os utilizadores conseguem aceder aos dados em qualquer altura e em qualquer lugar tornou-se uma das vantagens principal de ser uma tecnologia com demasiada procura atualmente. Nuvem, tournou-se assim numa das palavras que cada vez faz mais parte do nosso vocabulário diariamente.

Maioritariamente este projeto utilizou virtualização para conseguir simular uma rede local em que cada máquina tinha a sua função e todas se comunicavam entre si .Sem os conhecimentos adquiridos ao longo da licenciatura e da pós-graduação o objetivo do projeto nunca teria sido alcançado. Tiro também deste projeto uma experiência enriquecedora em termos consolidação de conhecimentos práticos como também, uma melhor perspetiva de como esta tecnologia funciona.

REFERÊNCIAS BIBLIOGRÁFICAS

Oracle.(s.d). *Brief History of Virtualization.* Consultado em 10 Fevereiro, 2018 em : https://docs.oracle.com/cd/E26996_01/E18549/html/VMUSG1010.html

Mueen Uddin ., Azizah Abdul Rahman ., Asadullah Shah E Jamshed Memon (2012). Virtualization Implementation Approach for Data Centers to Maximize Performance. *Asian Journal of Scientific Research,* p. 45.

IBM.(2007,Outubro). *Virtualization in Education.* Consultado em 10 Fevereiro, 2018 em: http://www-07.ibm.com/solutions/in/education/download/Virtualization%20in%20Education.pdf

TechTarget.(2010, Agosto). *What's the difference between Type 1 and Type 2 hypervisors?.* Consultado em 10 Fevereiro, 2018 em : http://searchservervirtualization.techtarget.com/feature/Whats-the-difference-between-Type-1-and-Type-2-hypervisors

TechTarget.(2016, Setembro). *SQL (Structured Query Language).* Consultado em 10 Fevereiro, 2018 em : https://searchsqlserver.techtarget.com/definition/SQL

Ross Mistry e Stacia Misner(Microsoft).(2014). *Introducing Microsoft SQL Server 2014 Technical Overview.* Consultado em 15 Fevereiro, 2018 em : https://download.microsoft.com/download/D/F/2/DF25A191-1FA4-4BC2-925C-492D616CF7FA/Microsoft_Press_ebook_Introducing_Microsoft_SQL_Server_2014_PDF.pdf

Douglas Laudenschlager e Olprod(Microsoft).(2017, Março). *Visão geral do ADO.NET.* Consultado em 15 Fevereiro, 2018 em : https://docs.microsoft.com/pt-br/dotnet/framework/data/adonet/ado-net-overview

Peter Mell e Tim Grance(NIST).(2011, Setembro). *The NIST Definition of Cloud Computing*. Consultado em 20 Fevereiro, 2018 em : https://csrc.nist.gov/publications/detail/sp/800-145/final

DXC.technology(s.d). *O Que É O Cloud Computing?*. Consultado em 20 Fevereiro, 2018 em : http://www.dxc.technology/pt/offerings/63346-o_que_%C3%A9_o_cloud_computing

Microsoft(s.d). *O que é computação em nuvem?*. Consultado em 20 Fevereiro, 2018 em : https://azure.microsoft.com/pt-br/overview/what-is-cloud-computing/

Dialogic(2017). *Introduction to Cloud Computing*. Consultado em 22 Fevereiro, 2018 em: https://www.dialogic.com/~/media/products/docs/whitepapers/12023-cloud-computing-wp.pdf

Rayne Wiselman, Jyothi Suri, Sudeep Kumar, Duncan Mackenzie(Microsoft).(2017, Julho). *What is Virtual Machine Manager?*. Consultado em 24 Fevereiro, 2018 em: https://docs.microsoft.com/en-us/system-center/vmm/overview?view=sc-vmm-1801

Anexos

Anexo 1 – *Form Login*

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

namespace Aplicacao
{
   public partial class login : Form
   {
      public login()
      {
         InitializeComponent();
         //Substituir os caracteres do texto da password por '*'
         txtPassword.PasswordChar = '*';
      }
      private void btnLogin_Click(object sender, EventArgs e)
      {
         //verificação dos campos
         if(txtUsername.Text != "" && txtPassword.Text != "")
         {
            //Ligação ao dominio
            string path = @"LDAP://DC.projetoglobal.local";
```

```
            string dominio = @"projetoglobal.local";
            string username = txtUsername.Text.Trim();
            string password = txtPassword.Text.Trim();
            string domUser = dominio + @"\" + username;
            //varificação autenticação dos dados inseridos
            bool loginSuccess = Authenticate(path, domUser, password);
            if (loginSuccess)
            {
               MessageBox.Show("Login Successful!");
               Form1 fAdmin = new Form1(txtUsername.Text);
               fAdmin.Show();
               this.Hide();
            }
            else
            {
               MessageBox.Show("Access Denied");
            }
         }
         else
         {
            MessageBox.Show("Please fill all the fields!");
         }
      }
      //Função da autenticação ao dominio
      private bool Authenticate(String path, String username, String password)
      {
         DirectoryEntry de = new DirectoryEntry(path, username, password,
AuthenticationTypes.Secure);
         try
         {
            DirectorySearcher ds = new DirectorySearcher(de);
```

```
            ds.FindOne();
            return true;
        }
        catch
        {
            return false;
        }
    }
  }
}
```

Anexo 2 – Form Form1 (Menu principal)

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Data.Entity;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;
using System.IO;
using System.Globalization;
using System.Text.RegularExpressions;
using iTextSharp.text;
using iTextSharp.text.pdf;

namespace Aplicacao
{
   public partial class Form1 : Form
   {    //Criação de um obejto
      user model = new user();
      public Form1(string username)
      {
         InitializeComponent();
         //Recolha do nome do utilizador que fez o login
         lblUsernameLogin.Text = "Welcome " + username + "!";
      }
      //Apagar todos os campos e deixar disponíveis alguns
```

```
void clear()
{
   txtFirstName.Text = txtLastName.Text = txtEmail.Text = txtCity.Text = "";
   imageView.Image = null;
   txtFirstName.Enabled = true;
   txtLastName.Enabled = true;
}
//Função para listar todos os utilizadores da base de dados
void DataGridView()
{
   dataGridViewUsers.AutoGenerateColumns = false;
   using (myDataBaseEntities db = new myDataBaseEntities())
   {
      dataGridViewUsers.DataSource = db.users.ToList<user>();
   }
}
//Meter indisponível os botões Eliminar e atualizar quando esta página é iniciada
private void Form1_Load(object sender, EventArgs e)
{
   clear();
   btnUpdate_field.Enabled = false;
   btnDelete_field.Enabled = false;
}
//Guarda os dados inseridos na base de dados
private void btnSave(object sender, EventArgs e)
{
   try
   {

      var teste = txtEmail.Text;
```

```
        if (txtFirstName.Text != "" && txtLastName.Text != "" && txtEmail.Text !=
"" && txtCity.Text != "" && imageView != null)
        {
            if (IsValidEmail(teste) == true)
            {
                model.firstName = txtFirstName.Text.Trim();
                model.lastName = txtLastName.Text.Trim();
                model.email = teste;
                model.city = txtCity.Text.Trim();
                model.image = ConvertFiltoByte(this.imageView.ImageLocation);
                using (myDataBaseEntities db = new myDataBaseEntities())
                {
                    db.users.Add(model);
                    db.SaveChanges();
                }
                clear();
                DataGridView();
                MessageBox.Show("User Saved");
            }
            else
            {
                MessageBox.Show("Please enter a valid email address!");
            }
        }
        else
        {
            MessageBox.Show("Please fill all the fields");
        }


    }
```

```
    catch (Exception ex)
    {
        throw new ApplicationException("Something went wrong...", ex);
    }
}
// Chama a função Clear para eliminar os dados
private void btnClear(object sender, EventArgs e)
{
    clear();
}
//Atualiza valores que foram inseridos para a base de dados
private void btnUpdate(object sender, EventArgs e)
{
    model.firstName = txtFirstName.Text;
    model.lastName = txtLastName.Text;
    model.email = txtEmail.Text;
    model.city = txtCity.Text;
    model.image = ConvertFiltoByte(this.imageView.ImageLocation);
    using (myDataBaseEntities db = new myDataBaseEntities())
    {
        db.Entry(model).State = EntityState.Modified;
        db.SaveChanges();
    }
    MessageBox.Show("Update Complete");
    DataGridView();
    clear();
    btnSave_field.Enabled = true;
    btnUpdate_field.Enabled = false;
}
//Elimina o registo secionado
private void btnDelete(object sender, EventArgs e)
```

```
    {
        if (MessageBox.Show("Are you sure want to delete?", "Delete",
MessageBoxButtons.YesNo) == DialogResult.Yes)
        {
            using (myDataBaseEntities db = new myDataBaseEntities())
            {
                var entry = db.Entry(model);
                if (entry.State == EntityState.Detached)
                {
                    db.users.Attach(model);
                }
                MessageBox.Show("User Removed");
                db.users.Remove(model);
                db.SaveChanges();
                DataGridView();
                clear();
                btnDelete_field.Enabled = false;
                btnUpdate_field.Enabled = false;
                btnSave_field.Enabled = true;
                txtFirstName.Enabled = true;
                txtLastName.Enabled = true;
            }
        }
    }
    //Deixa algumas textbox indisponíveis e duplo clique numa linha do gridview
passa os valores selecionados para as textbox
    private void dataGridViewUsers_DoubleClick(object sender, EventArgs e)
    {
        btnSave_field.Enabled = false;
        btnUpdate_field.Enabled = true;
        btnDelete_field.Enabled = true;
```

```
            txtFirstName.Enabled = false;
            txtLastName.Enabled = false;
            if (dataGridViewUsers.CurrentRow.Index != -1)
            {
                model.Id =
Convert.ToInt32(dataGridViewUsers.CurrentRow.Cells["Id"].Value);
                using (myDataBaseEntities db = new myDataBaseEntities())
                {
                    model = db.users.Where(x => x.Id == model.Id).FirstOrDefault();
                    txtFirstName.Text = model.firstName;
                    txtLastName.Text = model.lastName;
                    txtEmail.Text = model.email;
                    txtCity.Text = model.city;
                    imageView.Image = ConvertBytetoImage(model.image);
                }
            }
            btnClear_field.Enabled = false;
        }
        //Função para converter Byte em imagem
        private System.Drawing.Image ConvertBytetoImage(byte[] photo)
        {
            System.Drawing.Image newImage;
            using (MemoryStream ms = new MemoryStream(photo, 0, photo.Length))
            {
                ms.Write(photo, 0, photo.Length);
                newImage = System.Drawing.Image.FromStream(ms, true);
            }
            return newImage;
        }
        //Função para converter uma imagem em Byte
        private byte[] ConvertFiltoByte(string sPath)
```

```
        {
            byte[] data = null;
            FileInfo fInfo = new FileInfo(sPath);
            long numBytes = fInfo.Length;
            FileStream fStream = new FileStream(sPath, FileMode.Open, FileAccess.Read);
            BinaryReader br = new BinaryReader(fStream);
            data = br.ReadBytes((int)numBytes);
            return data;
        }
        //Botão para pesquisar imagem
        private void btnBrowse_Click(object sender, EventArgs e)
        {
            OpenFileDialog file = new OpenFileDialog();
            file.Title = "Please select a picture";
            file.Filter = "JPG|*.jpg|PNG|*.png|GIF|*gif";
            file.Multiselect = false;
            if (file.ShowDialog() == System.Windows.Forms.DialogResult.OK)
            {
                this.imageView.ImageLocation = file.FileName;
            }
        }
        //Botão para pesquisar por primeiro nome ou pelo último
        private void btnSearch_Click(object sender, EventArgs e)
        {
            dataGridViewUsers.AutoGenerateColumns = false;
            using (myDataBaseEntities db = new myDataBaseEntities())
            {
                string nomeSearch = txtSearch.Text;
                dataGridViewUsers.DataSource = db.users.Where(x =>
x.firstName.StartsWith(nomeSearch) || x.lastName.StartsWith(nomeSearch)).ToList();
            }
```

```
}
//Botão para listar todos os utilizadores
private void btnAllUsers_Click(object sender, EventArgs e)
{
   DataGridView();
}
//Aceitar no campo First Name só letras
private void txtFirstName_KeyPress(object sender, KeyPressEventArgs e)
{
   if (!char.IsControl(e.KeyChar) && !char.IsLetter(e.KeyChar))
   {
      e.Handled = true;
   }
}
//Aceitar no campo Last Name só letras
private void txtLastName_KeyPress(object sender, KeyPressEventArgs e)
{
   if (!char.IsControl(e.KeyChar) && !char.IsLetter(e.KeyChar))
   {
      e.Handled = true;
   }
}
//Aceitar no campo City só letras
private void txtCity_KeyPress(object sender, KeyPressEventArgs e)
{
   if (!char.IsControl(e.KeyChar) && !char.IsLetter(e.KeyChar))
   {
      e.Handled = true;
   }
}
//Verificação se o email é válido ou não
```

```
        private bool IsValidEmail(string strIn)
        {
            bool invalid = false;
            if (String.IsNullOrEmpty(strIn))
                return false;
            strIn = Regex.Replace(strIn, @"(@)(.+)$", this.DomainMapper);
            if (invalid)
                return false;
            return Regex.IsMatch(strIn,
              @"^(?("")(""[^""]+?""@)|(([0-9a-z]((\.(?!\.))|[-
!#\$%&'\*\+/=\?\^`\{\}\|~\w])*)(?<=[0-9a-z])@))" +
              @"(?(\[)(\[(\d{1,3}\.){3}\d{1,3}\])|(([0-9a-z][-\w]*[0-9a-z]*\.)+[a-z0-
9]{2,17}))$",
              RegexOptions.IgnoreCase);
        }
        private string DomainMapper(Match match)
        {
            bool invalid = false;
            IdnMapping idn = new IdnMapping();
            string domainName = match.Groups[2].Value;
            try
            {
                domainName = idn.GetAscii(domainName);
            }
            catch (ArgumentException)
            {
                invalid = true;
            }
            return match.Groups[1].Value + domainName;
        }
        //Extrair todos os dados que estão na base de dados para um PDF
```

```
private void btnPdfExport(object sender, EventArgs e)
{
    using (SaveFileDialog sfd = new SaveFileDialog() { Filter = "Pdf file|*.pdf",
ValidateNames = true })
    {
        if (sfd.ShowDialog() == DialogResult.OK)
        {
            iTextSharp.text.Document doc = new
iTextSharp.text.Document(PageSize.A4.Rotate());
            try
            {
                PdfWriter.GetInstance(doc, new FileStream(sfd.FileName,
FileMode.Create));
                doc.Open();
                PdfPTable table = new PdfPTable(6);
                PdfPCell cell = new PdfPCell(new Phrase("Users"));
                cell.Colspan = 6;
                cell.HorizontalAlignment = 1;
                table.AddCell(cell);
                table.AddCell("ID");
                table.AddCell("First Name");
                table.AddCell("Last Name");
                table.AddCell("Email");
                table.AddCell("City");
                table.AddCell("Image");
                using (myDataBaseEntities db = new myDataBaseEntities())
                {
                    var dataUser = db.users.Select(x => new
                    {
                        Id = x.Id.ToString(),
                        firstName = x.firstName,
                        lastName = x.lastName,
```

```
                    email = x.email,
                    city = x.city,
                    image = x.image
                }).ToList();
                foreach (var userInfo in dataUser)
                {
                    table.AddCell(userInfo.Id);
                    table.AddCell(userInfo.firstName);
                    table.AddCell(userInfo.lastName);
                    table.AddCell(userInfo.email);
                    table.AddCell(userInfo.city);
                    iTextSharp.text.Image img =
iTextSharp.text.Image.GetInstance(userInfo.image);
                    table.AddCell(img);
                }
            }
            doc.Add(table);
            doc.Close();
            MessageBox.Show("File Created");
        }
        catch (Exception ex)
        {
            MessageBox.Show(ex.Message, "Message", MessageBoxButtons.OK,
MessageBoxIcon.Error);
        }
        finally
        {
            doc.Close();
        }
    }
}
```

```
        }
        //Quando se fecha a página Form1, a página login aparece para recomeçar o
processo login
        private void Form1_FormClosed(object sender, FormClosedEventArgs e)
        {
            login lg = new login();
            lg.Show();
        }
    }
}
```