INSTITUTO SUPERIOR DE TECNOLOGIAS AVANÇADAS

PROJECTO GLOBAL

RELATÓRIO FINAL

RICARDO JORGE COSTA MORAIS

3º ANO LICENCIATURA EM INFORMÁTICA

ALUNO Nº 8131 (Licenciatura) / A008 (Pós-Graduação)

Turma B – Diurno

TRABALHO REALIZADO COM A ORIENTAÇÃO DE:

PROFESSOR DOUTOR PEDRO BRANDÃO

PROFESSORA MARIA INÊS

12 DE JUNHO DE 2018

LISBOA

Resumo

A tecnologia de Virtualização e o conceito de Cloud Computing têm crescido a um nível exponencial nos últimos anos. O Cloud Computing surgiu na primeira década do século. XX1, mas a ideia por detrás da mesma (centralizar a computação e armazenados de dados em *datacenters*) começou a ser pensada há muito tempo.

A virtualização é uma tecnologia que é usada para partilhar as capacidades de computadores físicos dividindo os recursos entre os sistemas operacionais oferecendo inúmeras vantagens a uma infraestrutura de TI (com uma melhor utilização dos recursos e redução do poder de consumo) através da máquina virtual.

Tanto o SQL como o ADO.NET são duas tecnologias importantes. O SQL, linguagem padrão para gerir e consultar (manipular) uma base de dados e o ADO.NET é um conjunto de classes que vai interligar (através de código) com o SQL.

Palavras-Chave: Virtualização, Nuvem, Computação na Nuvem, Tecnologia, Inovação, SQL, ADO.NET

Abstract

The Technology of Virtualization and the concept of Cloud Computing has grown up last years. The Cloud Computing emerged in the first decade of 21st but the ideia behing it (centralizing computation and storage in distributed data centres maintained by third party) began to be thought of in the 1990s.

The Virtualization is a Technology used to share the capabilities of physical computers by splitting the resources among OSs offering a lot of advantages to a IT Infrastructure (such as better resource utilization and reduction of power comsumption) through a Virtual Machine.

Both SQL and ADO.NET are two important technologies. SQL, the standard language for managing and querying (manipulating) a database, and ADO.NET is a set of classes that will connect (through code) with SQL.

Key Words: Virtualization, Cloud, Cloud Computing, Technology, Innovation, SQL, ADO.NET.

Índice

Introdução

O presente relatório tem como principal objectivo, numa primeira parte, efectuar uma apresentação das tecnologias de virtualização, *cloud computing*, SQL e ADO.NET mostrando as principais características e funcionalidades de cada uma destas tecnologias, bem como o impacto que têm e vão continuar a ter no mundo chamado de Internet. Noutro lado, uma parte de desenvolvimento em que é explicado, ao detalhe, como foi construída e desenvolvida uma aplicação em *Windows Form* - um *CRUD* que tinha como principal objectivo a utilização de máquinas virtuais, todas em comunicação entre si, o SQL para a criação de uma base de dados que irá disponibilizar a informação sobre as publicações e a tecnologia ADO.NET para ser feito o acesso à base de dados.

O relatório final encontra-se organizado em quatro capítulos: o primeiro é sobre o Estado da Arte em que é realizado um estudo bibliográfico das tecnologias que já foram referenciadas, nomeadamente as vantagens da sua utilização e a sua importância e impacto no presente e no futuro; o segundo capítulo é a Contextualização, em que é realizado um contexto sobre o tema em si, uma terceira parte de Desenvolvimento, que como já foi referido, serve para explicar o código da aplicação que foi realizada em *Windows Form* e, por último, o capítulo de discussão que em que se faz uma minuciosa análise de todo o relatório realizado.

Estado da Arte

Cloud Computing

O *Cloud Computing*, ou, em português, computação na nuvem, é o fornecimento de serviços de computação (servidores, armazenamento, rede, software e outros serviços) independentemente do local onde se encontram os utilizadores, (Rosa, 2013, p. 5).

A tecnologia evolui de forma constante e a estrutura *cloud computing* foi uma das que mais cresceu nos últimos anos devido ao que a mesma oferece. Apesar de o conceito de *cloud computing* ter sido idealizado no século XX, mais concretamente no início da década de '60 por intermédio de John McCarthy sugeriu que a computação fosse oferecida como um serviço público e um pouco mais tarde, em 1962, por Joseph Carl Robnett Licklider, um físico que conseguiu que as pessoas partilhassem dados de forma global e com essa descoberta criou a rede *Arpanet*, conhecida globalmente como a Mãe de toda a Internet. O objectivo de Licklider era unir as bases militares com os departamentos de pesquisa do governo americano, (IPM, 2017).

Mesmo com os grandes avanços de McCarthy e Licklider, conhecidos como os grandes pioneiros da computação na nuvem, foi só apenas nos anos 90 é que se deu um avanço no que diz respeito à computação na nuvem. (ibid).

A nuvem é uma representação para a internet entre componentes arquiteturais, baseada numa abstração que oculta à complexidade da infraestrutura. Cada parte desta infraestrutura é provida como um serviço, e estes serviços são normalmente alocados em *datacenters*, utilizando hardware partilhado para computação e armazenamento, (Sousa, 2009).

Com todo o natural crescimento que esta tecnologia teve, a mesma começou a ser oferecida comercialmente em 2008, quando as empresas (primeiramente a Amazon) gradualmente adoptaram este serviço e desta então a "nuvem" nunca mais parou de crescer tendo inúmeras plataformas de Cloud nos dias de hoje e prevê-se que o investimento mundial ultrapasse os 100 bilhões de dólares já em 2018, (Brinda & Heric, 2017)

A evolução do *Cloud Computing* pode ser vista como uma inovação em diversos sentidos. De uma perspectiva tecnológica é um avanço enorme para a computação, aplicando conceitos de virtualização para utilizar o *hardware* com maior eficiência. No entanto, um ponto de vista diferente é analisar a computação na nuvem a partir de uma perspectiva de implantação de TI. Nesse sentido, a computação na nuvem tem o potencial de revolucionar o modo como

os recursos e aplicações de computação são fornecidos, quebrando cadeias tradicionais de valor e abrindo espaço para novos modelos de negócios, (Böhm & Krcmar, 21).

De acordo com o NIST (National Institute of Standards and Technology, s. d.), o *cloud computing* é um modelo que permite o acesso ubíquo, conveniente e a pedido, via rede a um conjunto de recursos de computação partilhados que podem ser rapidamente libertados e fornecidos, com um mínimo esforço e sem interacção com o fornecedor (chamado de *self-service*).

Numa *cloud* podemos disponibilizar os ficheiros pretendidos, como aplicações de *software*, espaço para armazenamento de ficheiros ou uma base de dados. Todos estes serviços, são disponibilizados através da Internet e caso o cliente tenha acesso a essa *cloud*, pode ver tudo o que lá foi publicado, independentemente do computador, sistema operativo distintos. E é possível aceder em qualquer parte do globo e mantendo uma experiência de utilização comum quando acedem de um dispositivo (computador, *smartphone* ou *tablet*, por exemplo) diferente, (Rosa, 2013, pp. 5-6).

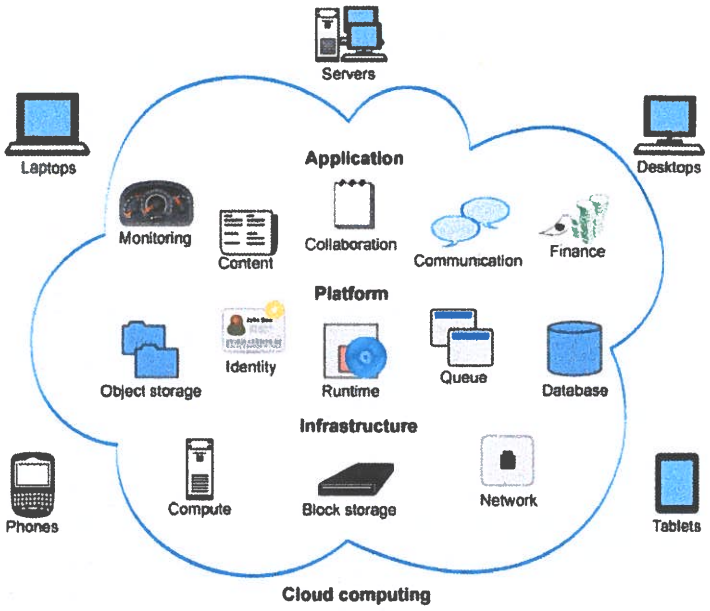

*Figura 1. Configuração de uma cloud. **Fonte:** tihousesoft (s.d)*

Características do Cloud Computing

O *cloud computing* é composto por cinco características principais: o *on-demand self-service* – o consumidor pode usar os serviços da nuvem, aumentando ou diminuindo os recursos, sem exigir qualquer interacção humana com o fornecedor de serviços. O *broad network access* – Em português, significa um amplo acesso à rede, ou seja, significa que os serviços disponibilizados na nuvem podem ser acedidos através de qualquer dispositivo (telemóveis, computadores, etc). O *resource pooling* que permite aos fornecedores da nuvem agrupar os recursos de TI em larga escala para atender múltiplos consumidores. Como clientes diferentes utilizam *Cloud Servers* diferentes, o servidor físico subjacente e optimizado, ou seja, há menos desperdício de recursos de memória RAM, utilização de processador, de disco e energia., A *rapid elasticity* – possuí elasticidade. É possível aumentar ou diminuir a capacidade de recursos quando o utilizador assim entender. Vejamos, num dia o utilizador poderá necessitar de apenas um servidor, mas na semana a seguir poderá precisar de cinco, e assim o utilizador poderá aumentar a capacidade de forma instantânea. E o *measured service* – todos os serviços são controlados automaticamente pela *cloud*. O fornecedor de serviço monitoriza a prestação de serviços por diversas razões mantendo assim a transparência para o consumidor. É uma ajuda para ambos, porque assim o consumidor pode optimizar a sua utilização na *cloud* e ajuda o fornecedor na cobrança de recursos. É um modelo de pagamento após utilização, não de pré-pagamento, (Pinheiro, 2010).

Modelos de Implementação

Os modelos de implementação (qual será a estrutura onde a *Cloud* irá ser implementada) são quatro, em que se pode dividir entre dois principais (*Cloud* Privada e Pública) ou se preferir uma solução mista (*Cloud* em Comunidade e *Cloud* Híbrida).

Uma *cloud* pública é uma nuvem em pode ser acedida em toda a Internet. É o tipo de nuvem mais barato, pois os custos de *hardware*, aplicações e largura de banda são cobertos pelos fornecedores. A empresa apenas paga pela capacidade a utilizar e a *cloud* privada é, ao contrário da nuvem pública, é aquela que fica dentro do ambiente da empresa (*firewall*) e tem o acesso restrito geralmente apenas aos funcionários dessa mesma empresa e aos seus parceiros. É importante para manter segurança. Uma empresa que exige segurança nas suas informações deverá optar por uma rede privada ao invés de uma rede pública, (Pinheiro, 2010).

Se o cliente tem o objectivo de usar uma rede privada e uma rede pública na sua organização, é chamado de *cloud* híbrida. Uma *cloud* híbrida é composta por duas ou mais

clouds de tipo pública e privada, permitindo que os dados e as aplicações sejam partilhados. A utilização de uma *cloud* híbrida permite às empresas dimensionarem os recursos informáticos e também elimina a necessidade de gastos excessivos de capital para lidar com picos de procura a curto prazo, (Azure, s.d.).

O quarto tipo de *cloud* é a *Community Cloud*. É partilhada por várias organizações que partilham interesses mútuos (política, jurisdição, etc). O modelo pode ser administrado pelas próprias empresas ou por terceiros, além disso, a infraestrutura pode ser interna ou externa às empresas, (Pinheiro, 2010).

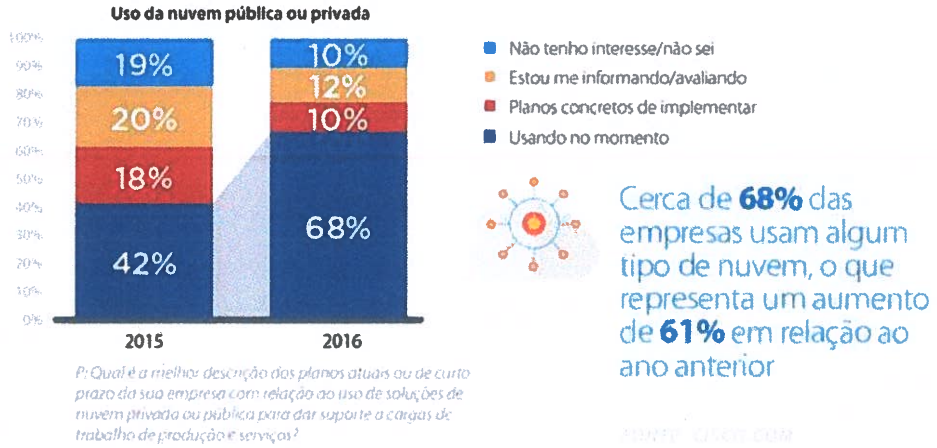

Figura 2. *O crescimento da cloud em um ano.* ***Fonte:*** *MasterD (s.d)*

Serviços de Modelo

Quanto aos serviços de modelo, o *cloud computing* é dividido em três grandes grupos dependendo do à vontade que o cliente tem com a infraestrutura: a *Infrastructure as a Service (IaaS)* que é a utilização de recursos de infraestruturas básicas de computação e armazenamento. Por exemplo, é a disponibilização de máquinas virtuais para o cliente, com recursos de processamento, armazenamento de dados, servidores e componentes de rede. É usado por administradores da rede. O *Platform as a Service (PaaS)* é fornecido toda a plataforma e ambiente de desenvolvimento para o cliente. O cliente paga ou aloca somente os recursos necessários, sem desperdício de *hardware* ou limites de processamento, para o desenvolvimento de seus *softwares*, é tipicamente utilizador por programadores (Pinheiro, 2010) e o *Software as a Service (SaaS)*, se o *IaaS* disponibiliza os recursos de *hardware*, o *Software as a Service* apenas vemos disponível os de software. Ou seja, o serviço é a disponibilização de uma aplicação de e-mail (Gmail, por exemplo). É vantajoso para o cliente não profissional porque o mesmo não tem de se "preocupar" com a manutenção do software

bem como algumas actualizações, ficando esse "trabalho" a cargo do fornecedor da *cloud*. Ambos os lados são favorecidos. O fornecedor não se preocupa tanto em vender o software, mas em aprimorá-lo, atualizá-lo. Isso faz com que o cliente tenha sempre um serviço bom sem se preocupar com gastos extras, (Espojeira, 2013).

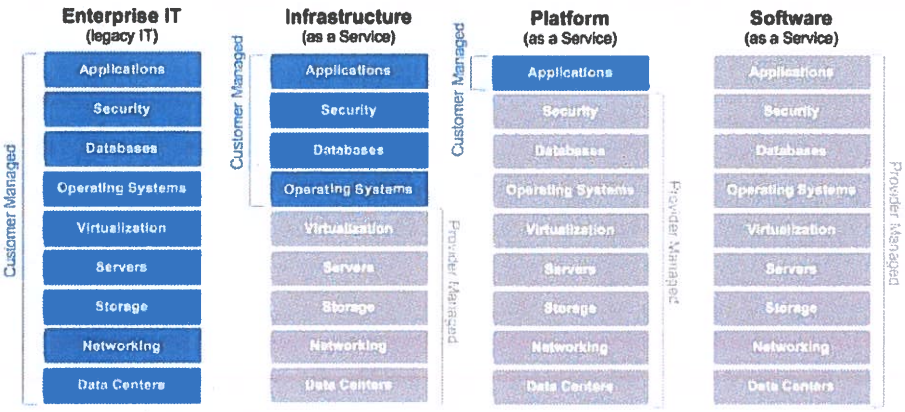

Figura 3. Um comparativo entre os diferentes serviços de modelo. ***Fonte:*** *Microsoft (s. d.)*

Vantagens do Cloud Computing

As cinco características, os quatro tipos de nuvem existentes e os três serviços de modelo disponíveis para o cliente são o melhor que a *cloud* são grandes vantagens para a utilização de uma cloud seja qual foi o tipo, mas existem outras vantagens tais como: O modelo de computação na nuvem foi desenvolvido com o objectivo de fornecer serviços de fácil acesso, baixo custo e com garantias de disponibilidade e escalabilidade. Este modelo visa fornecer, basicamente, três benefícios. O primeiro benefício é reduzir o custo na aquisição e composição de toda infraestrutura requerida para atender as necessidades das empresas, podendo essa infraestrutura ser composta sob demanda e com recursos heterogêneos e de menor custo. O segundo é a flexibilidade que esse modelo oferece no que diz respeito à adição e substituição de recursos computacionais, podendo escalar tanto em nível de recursos de *hardware* quanto *software* para atender as necessidades das empresas e usuários. O último benefício é prover uma abstração e facilidade de acesso aos usuários destes serviços. Neste sentido, os usuários dos serviços não precisam conhecer aspectos de localização física e de entrega dos resultados destes serviços, pois é possível aceder à sua nuvem em qualquer lado e em qualquer altura, desde que tenha um dispositivo com conexão à Internet. (Sousa, Machado, & Moreira, 2011).

Também traz outras grandes vantagens como a sustentabilidade, a possibilidade de promover a sustentabilidade de uma empresa a nível económico e ambiental. Por exemplo,

gastar menos na compra de manutenção de equipamentos, reduzir a dimensão da sua infraestrutura TI e optimizar o espaço físico. O investimento na Infraestrutura será obviamente uma das grandes vantagens porque se uma empresa optar por investir numa cloud, não tem de se preocupar, como já foi referido, com aquisição, manutenção, actualização de servidores, licenças, espaço de armazenamentos, cabos partidos, etc. Com um contrato, tem facilmente acesso a discos virtuais, base de dados, aplicações de ambiente de trabalho e tudo o que for necessário para automatizar, centralizar e optimizar o negócio específico da empresa, (Cloud21, 2016)

A nível de segurança, pode existir algum questionamento sobre se é mais ou menos segura que o modelo tradicional. Mas com cada vez mais especialistas no assunto, os fornecedores de serviços de *cloud*, estão sistematicamente actualizados em relação às melhores práticas de segurança e mantém altos níveis de serviços para se manterem competitivos. Existem muitas formas de assegurar a segurança na nuvem, como o mais habitual com os *backups*, a garantia de recuperação de dados em casa de acidente (disaster recovery) e muitas outras alternativas para manter os dados em segurança, (Zanutto, s.d.).

Por último, e não menos importante, é o impacto ambiente. A utilização de uma *cloud*, reduz o consumo de *hardware*, as emissões de dióxido de carbono e os custos de energia em qualquer momento, os servidores são usados de acordo com o requisito e isso economiza bastante energia, (GrupoCloud, s.d.)

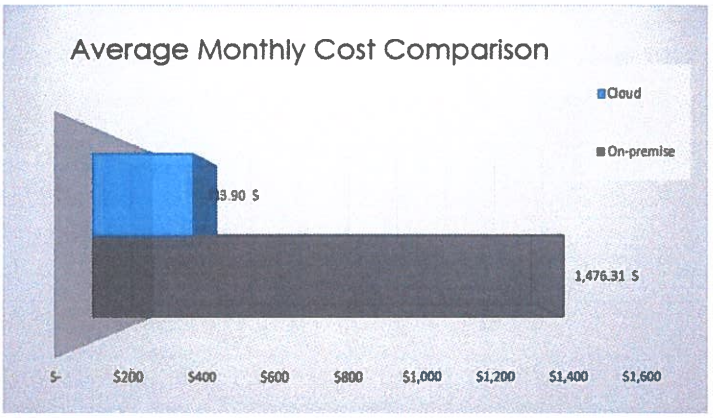

Figura 4. *Um comparativo onde é evidente a diferença de custo entre ter uma cloud e um servidor tradicional.* ***Fonte:*** *SherWeb (s.d.)*

Virtualização

De acordo com a Microsoft (s.d.), a Virtualização «*cria um ambiente informático simulado ou virtual, ao invés de um ambiente físico* [...] *Isto permite às organizações particionarem um único computador ou servidor físico em várias máquinas virtuais*», ou seja, ao criar vários recursos através de um único computador ou servidor, a virtualização melhora a estabilidade e as cargas de trabalho, resultando numa utilização de menos serviços gerais, menor consumo de energia e manutenção da infraestrutura.

A virtualização é, portanto, uma técnica que permite partilhar e utilizar recursos dum único sistema computacional em outros (máquinas virtuais). Ou seja, cada máquina virtual oferece um sistema computacional completo muito similar a uma máquina física. E tudo isto, leva com que cada máquina virtual possa ter o seu sistema operativo e possível conectar em rede cada uma dessas máquinas virtuais, (Alecrim, 2012).

Desde 2008, com o lançamento do Windows Server 2008, a Microsoft incluiu no seu sistema operativo a tecnologia de virtualização *Hyper-V* – tecnologia usada na pós-graduação, suportada quer em instalações *server core* quer em instalações completas, (Rosa, 2013, pp. 3-4).

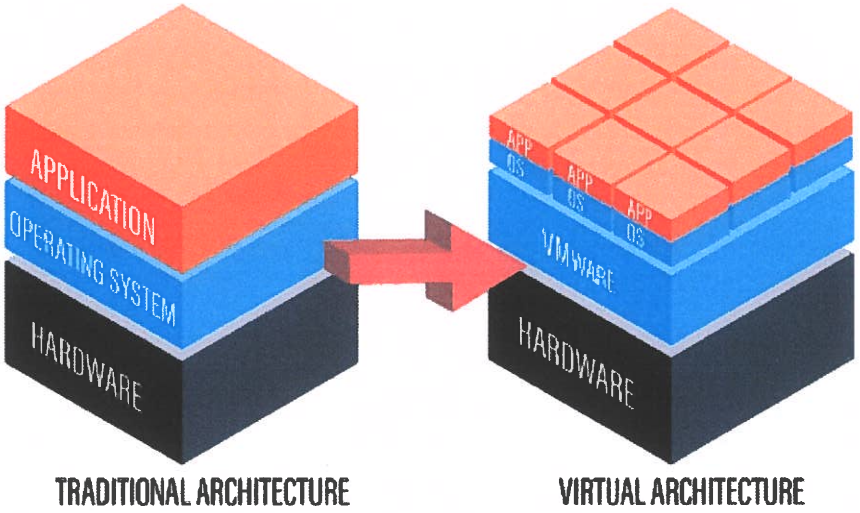

Figura 5. *A diferença entre o modelo tradicional e o modelo de virtualização.* ***Fonte:*** *Cybertrol Engineering (s.d.)*

Benefícios da utilização da Virtualização

São muitos os benefícios ao utilizar a virtualização e nos últimos anos tem existindo um crescimento exponencial no uso da virtualização comparativamente ao uso da arquitetura dita tradicional)

A virtualização oferece a possibilidade de administradores de TI destinarem recursos conforme a estratégia e necessidade do negócio, em vez de atribuir uma única tarefa estática e que acaba por ficar "presa" no *hardware*. (Monginho, 2012).

Entre as muitas vantagens, aquela que é considerada principal é a economia, pois ao utilizar a máquina virtual, o cliente tem diversas vantagens financeiros, por exemplo, no corte de aquisições de novos *hardwares*, gastos com instalação, manutenção, refrigeração, entre outros que possa a vir custa algum dinheiro extra ao cliente, (Alecrim, 2012).

Em empresas que precisam de mais do que um servidor para executar os seus sistemas, a virtualização traz uma redução de custos de aquisição de solução, pois um mesmo servidor pode ser utilizado por mais do que um sistema operativo, e mesmo que precise de comprar um servidor com maior capacidade, o custo extra desses recursos é sensivelmente menor comparativamente a comprar um novo servidor para cada sistema, (ibid).

Além de reduzir os custos de aquisição e de energia, a virtualização tem impacto directo no meio ambiente. Isto porque, no final de vida útil, existirão menos servidores para serem descartados, resultando em menos materiais tóxicos no meio ambiente. Num ambiente de virtualização, é possível configurar para que alguns *hosts* sejam desligados durante o período de menor demanda (como aos fins-de-semana e a noite, por exemplo), (Fernando, 2014).

Outra principal vantagem é a independência do fornecedor porque devido à abstracção total da parte física, a máquina virtual não fica dependente de um tipo de processador, placa de rede ou qualquer outro dispositivo, podendo assim migrá-la de *hardware* facilmente, (Fernando, 2014). E essa independência é possível através de um gerenciamento simplificado porque fornece uma plataforma de gestão interativa, intuitiva e facilitada ao cliente para acelerar ou automatizar a administração de armazenamento e tarefas de *backup* em ambientes virtualizados. (InforTrend, s.d.)

E se existir algum problema com a máquina virtual, não existe grande preocupação porque é possível fazer *backups* bem como o *disaster recovery*. Um *cluster* de servidores virtuais traz por padrão a protecção contra a falha física de servidores, as máquinas virtuais são

protegidas pelo *High Avaliability*, onde, na falha do servidor que está a executar, a mesma é ligada num outro servidor automaticamente, e em poucos minutos o sistema está recuperado automaticamente. Portanto, a virtualização permite que as empresas executem as suas políticas de *backup* de dados com agilidade e precisão. Ou mesmo coma utilização do *Hyper-V*, existe a possibilidade de agendar backups para acontecer a um determinado dia da semana em que é possível fazer o *backup* das máquinas virtuais que o cliente bem entender (as mais fundamentais/importantes na perspectiva do cliente), assim tem a certeza que se algum problema ocorrer, tem a sua máquina devidamente protegida, (Fernando, 2014).

A segurança, segundo Neil MacDonald (2011), as máquinas virtuais acabam por apresentar menos segurança que as físicas devido ao *hypervisor*. Se o sistema operativo *host* estiver vulnerável a algum tipo de ataque, todas as máquinas virtuais que estiverem aí hospedadas ficam igualmente vulneráveis a ataques visto que o VMM não é mais que uma camada de *software*, e como qualquer *software*, está sujeito a vulnerabilidades e a ataques. Porém, com o avançar desta mesma tecnologia e com cada vez mais profissionais especializados nesta área, essa vulnerabilidade actualmente já não se confirma tanto, (ComputerWorld, 2018).

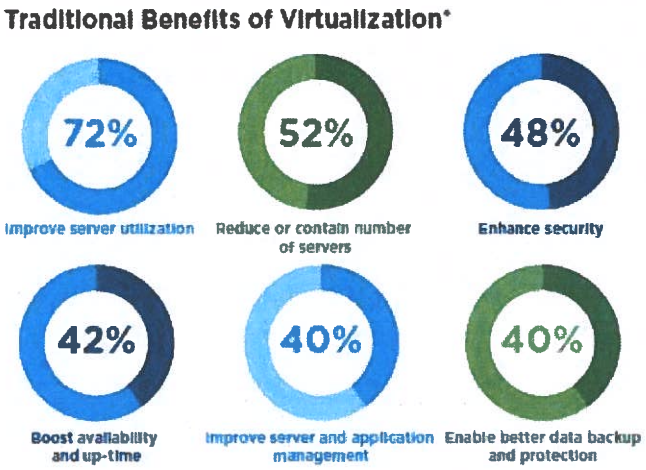

Figura 6. Os principais benefícios da Virtualização de acordo com um estudo patrocinado pela VMware. **Fonte:** *GovmLab (s.d.)*

Tipos de Virtualização

Existem três tipos de Virtualização, fala-se em concreto da Virtualização de Servidores, Virtualização de *Desktop* e, por último, a Virtualização de Aplicações, (VMware, s.d.).

A virtualização de servidores, consiste em rodar vários sistemas operativos na mesma máquina. Isto é possível devido ao uso de programas específicos (por exemplo, o VMware) que geram máquinas virtuais. Estas simulam os componentes físicos de um computador, possibilitando que um sistema operativo diferente seja instalado em cada uma delas. Cada máquina virtual criada é um ambiente operacional completo, seguro e totalmente isolado como se fosse um computador independente. Este tipo de virtualização é, actualmente, a mais usada, (Rocha, 2013)

Os principais benefícios da utilização de virtualização de servidores: possibilita a consolidação de servidores físicos, derrubando os custos de operação de *datacenters*; redução de equipamentos, como os *switches*, cablagem, equipamentos de refrigeração do ar condicionado, etc; Elimina a necessidade de janelas para manutenção de servidores, normalmente causadas por problema físicos ou geradas pela necessidade de *upgrade*; e contribui significativamente para a redução de emissão de dióxido de carbono ajudando assim, em parte, para uma redução do aquecimento global. A virtualização de servidores também aumenta a velocidade de implantação de cargas de trabalho, melhora o desempenho de aplicativos e aumenta a disponibilidade, (VMware, s.d.).

A *virtualização de desktop* é parecida com a de servidores, ou seja, possuir a capacidade de executar vários *desktops* virtuais num ou mais servidores. Na virtualização de *desktops* é adoptado o modelo cliente-servidor, onde todos os programas, aplicações, processos e dados são mantidos e executados de uma forma centralizada. Ou seja, na virtualização de *desktop*, cada utilizador possui o seu próprio sistema operativo e as suas aplicações, tal como se estivesse a utilizar um ambiente de trabalho normal. (ibid).

Os principais benefícios da utilização da virtualização de *desktop*: como já foi referido, os dados são mantidos e executados de forma centralizada; baixo custo para implementar novas aplicações; e o controlo eficiente dos dados, mantendo-os seguros dentro do *datacenter* da empresa; e proporciona uma experiência de utilizador remoto avançada, semelhante à proporcionada pelas aplicações em execução num PC local, (LearningSolutions, 2017).

O terceiro e último tipo de virtualização é a virtualização de aplicações onde as aplicações executadas na máquina local ou virtual, utilizam os seus recursos, mas não tem permissões para fazer qualquer tipo de alteração. Ao invés disso, são executadas num pequeno ambiente virtual que contém as entradas dos registos, arquivos, *DLLs* e os demais componentes que são necessários para executar essas aplicações. Este ambiente virtual age como uma camada entre aplicação e o sistema operativo, (VirtueIT, 2015)

Os principais benefícios da utilização da virtualização de aplicações: permite que se use ao máximo os recursos de hardware sem comprometer o desempenho, torna as aplicações mais móveis e faz com que o processo de migração ocorra com mais facilidade do que em qualquer servidor no ambiente, além de adicionar outra camada de disponibilidade de linhas de base, (VMware, s.d.); Simplifica o processo de upgrade numa aplicação. É preciso apenas virtualizar a aplicação e disponibilizar para os clientes; não existe qualquer preocupação se a aplicação é compatível com a nova versão do *Windows*, a virtualização de aplicação garante isso e não há necessidade de instalação individual de *software* em cada *desktop*, as aplicações não são mais instaladas, mas sim executadas, (VirtueIT, 2015)

Tipo de Hypervisor

O hypervisor é uma cada de software localizada entre o hardware e as máquinas virtuais, e é responsável por fornecedor os recursos da máquina física para a máquina virtual. Os *hypervisores* podem ser executados em dois tipos de virtualização:

O *hypervisor* em tipo 1 (fig. 9), está a ser executado directamente no hardware físico do host e tem como objectivo controlar o hardware e gerir sistemas operativos "guests". É um tipo de virtualização importante porque nesta camada, é possível gerir os recursos para as máquinas virtuais, bem como alocação da memória, (Costa, 2009).

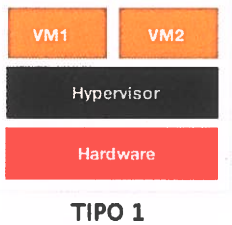

Figura 7. *Hypervisor do Tipo 1.*
Fonte: *LEMAF Blog.ti (s.d.)*

Caso o hypervisor esteja em tipo 2 (fig. 109, está a ser executado sobre um sistema operativo. Como se pode ver na figura 10, existe uma camada adicional do Sistema Operativo em relação ao do tipo 1 e usa mais recursos da máquina, (ibid).

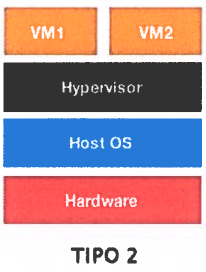

Figura 8. *Hypervisor do Tipo 2.*
Fonte: *LEMAF Blog.ti (s.d)*

SQL

O *Structured Query Language*, ou simplesmente SQL, foi desenvolvido para ser uma linguagem padrão para realizar operações numa base de dados, (Halvorsen, 2016, p. 5). É uma linguagem tão importante que Donald Chamberlin (s.d) afirma que «*O padrão SQL tem sido útil para fornecer um mecanismo para a evolução controlada da linguagem*».

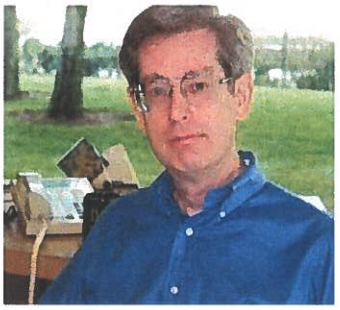

Figura 9. *Donald Chamberlin (na imagem) e Raymond Boyce foram os criadores do SQL.* ***Fonte:*** *Alchetron (s.d)*

A primeira versão do SQL foi chamada de SEQUEL (Structured Query English Language), surgi no ano de 1974 pela IBM, num dos seus laboratórios na Califórnia. Entre 1976 e 1977, a SEQUEL foi revisada tendo o seu nome sido alterado para o nome que ainda é conhecido nos dias de hoje – SQL – devido a SEQUEL ser uma marca registada de uma empresa de aeronaves, (Dybka, 2014)

O sucesso foi tão grande que acabou sendo padronizada por entidades como a ANSI – American National Standards Institute e pela ISSO – Internacional Standard Organization no final dos anos 80, a fim de unificar um padrão de linguagem para operações numa base de

dados, mas ao longo dos anos, o SQL possuí muitas variações e extensões produzidas por diferentes fabricantes de base de dados, (Chamberlin, 2012, p. 80).

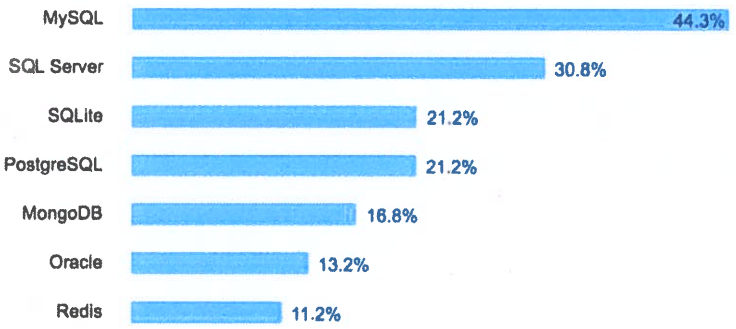

Figura 10. *As variantes de SQL surgem nos 4 primeiros lugares num estudo 2017.* ***Fonte:*** *Stack Overflow (s.d.)*

Com toda a naturalidade, o SQL passou a ser uma das linguagens de programação mais usadas do mundo muito devido à sua simplicidade e facilidade de uso. E porque nos dias de hoje, a quantidade de dados que uma organização tem de manipular e organizar é cada vez maior e é impensável que isso aconteça manualmente. Assim, torna-se mais fácil encontrar a informação numa base de dados que recorra a uma das tecnologias de informação, (Endeavor, 2015).

Vantagens em utilizar o SQL

As vantagens em usar esta linguagem de programação são imensas, e por isso é que se tornou uma linguagem tão popular e tão usada no mundo da programação. Eis as principais vantagens: O SQL tem uma alta velocidade – as *queries* podem ser usadas para recuperar grandes quantidades de registos de uma base de dados de forma rápida e eficiente e o facto de não precisar de código (porque tem ambiente gráfico) torna muito fácil a sua utilização, é portátil, o SQL "corre" em computadores, tablets, telemóveis, servidores. Uma *database* SQL pode ser movida de um dispositivo para o outro sem qualquer tipo de problema. É um tipo de programação muito pretendido. Muitos trabalhos – desenvolvimento web ou análises de dados, requerem habilidades em SQL, (Thakur, What is DMBS, 2017).

Instruções Principais em SQL

A linguagem SQL é dividida em vários grupos e cada um com o seu conjunto próprio de instruções: O DML – *Data Manipulation Language* – que permitem manipular informações na base de dados através dos Comandos Insert (inserir um registo), Update (alterar um registo) e Delete (apagar o registo criado), (Zaïane, 2016) o DCL – *Data Control Language* – as instruções SQL da SLC controlam aspectos destinados à autorização de dados e licenças de

utilizadores para manipulações de dados através dos seus comandos de Grant (para atribuir privilégios) e Revoke (retirar privilégios), (Chapple, 2017), o DDL – *Data Definition Language* – os comandos DDL, oferecem a manipulação das tabelas através do Create (criar base de dados, objectos), Alter (modificar os objectos na base de dados), Drop (elimina objectos na base de dados), Rename (mudar o nome de um objecto), Truncate (elimina todos os registos de uma tabela) e Comment (possibilidade de adicionar comentários à tabela), (Google, 2018), o DQL – *Data Query Language* – É uma linguagem de consulta de dados et em apenas um comando, mas é dos mais utilizados, o comando Select (que permite selecionar dados pertencentes às tabelas), (BD, 2013) e a TCL – *Transaction Control Language* – as mudanças que foram feitas pela DML, as instruções SQL da TCL permitem gerir a mudança. Os comandos utilizados são o Commit (finaliza uma transacção), Rollback ("esquece" por completo os dados existentes no último Commit) e o SavePoint (identifica um ponto de transacção para mais tarde efectuar-se um Rollback se necessário), (Chapple, 2017) .

ADO.NET

De acordo com a própria Microsoft (s.d.), ADO.NET não é mais que «*um conjunto de classes que expõem serviços de acesso a dados para desenvolvedores do .NET Framework* [...], *fornece um conjunto rico de componentes para criar aplicações distribuídos e de partilha de dados.*».

O ADO.NET é uma evolução do modelo ADO mas sem reaproveitar a tecnologia do mesmo. A sua única herança é o nome, e mesmo assim somente a sua abreviação, pois ADO.NET não significa ActiveX Data Objects.NET e sim, simplesmente, ADO.NET, (Húngaro, 2015).

As principais características desta versão do ADO.NET em relação às anteriores é que passou a ser possível trabalhar desconectado a partir da base de dados usada, tem uma forte integração com XML e ASP.NET e também, o uso do ADO.NET é independentemente de qualquer linguagem de programação que seja utilizada, (Besteiro & Rodriguez, 2015, p. 3).

Neste caso em concreto, o ADO.NET será importante porque fornece um acesso consistente à base de dados – através do SQL. Todas as aplicações podem utilizar o ADO.NET para estabelecer ligações a essas fontes de dados de modo a recuperar, manipular e actualizar os dados. Portanto, é justo dizer que o ADO.NET é um conjunto de classes para trabalhar com dados, (Ferreira, 2004, p. 5).

O acesso a dados com o ADO.NET pode ser de duas formas, conectado ao banco de dados (DataReader) ou desconectado (DataSet – mantém uma cópia de base de dados na memória), (Reilly, 2005).

As classes do ADO.NET trabalham juntas para fornecer o acesso a dados tabulares. A biblioteca inclui dois grandes grupos de classes: aqueles que administram os dados e aqueles que se comunicam com os dados externos, (Húngaro, 2015).

Principais objectos do ADO.NET

O ADO.NET possuí diversos objectos na sua biblioteca, mas os que mais se destacam são os objectos de Connection – o objecto *connection* serve para fazer a conexão à base de dados. Para estabelecer essa mesma ligação, é necessário a string de conexão, o *ConnectionString*; o objecto Command, se o Connection permitia fazer a ligação à base de dados, então o objecto Command é utilizado para executar comandos em SQL (as chamadas *queries*) contra a base de dados a que foi feita a conexão, os métodos para executar esses comandos dividem-se em três, o Command ExectureReader – os Comandos SQL que retornam dados através do Select, o Command ExecuteNonQuery – Comandos SQL que não retornam dados (Insert, Update, Delete) e o Command ExecuteScalar – que retorna um valor único, ou seja, retorna o primeiro valor depois de ser executado um Select (Através da propriedade Max, Count); como principal objecto existe ainda o DataSet, que, como já foi referido, é uma representação em memória da base de dados com uma ou mais tabelas (datatables). Permite receber os dados de uma base de dados e trabalhar com eles mesmo estando desligado da base de dados; O objecto DataTable representa uma tabela, contida num DataSet. Tem colunas (datacolumns) e ainda linhas (datarows); O DataReader, é um objecto utilizado para receber os resultados de um ExecuteReader do Command e permite aceder a todas as colunas e linhas da tabela com o método Read() e, por último, o objecto DataAdapter – é utilizado para "preencher" o objecto DataSet. Através do método Fill(), o DataAdapter copia os dados da base de dados para o DataSet, permitindo assim o trabalho desconectado da base de dados, quando se realiza alterações nessa "cópia" da Base de Dados, o método Update deste objecto (DataAdapter) efectua ligações na base de dados original. (Kaupa, 2014).

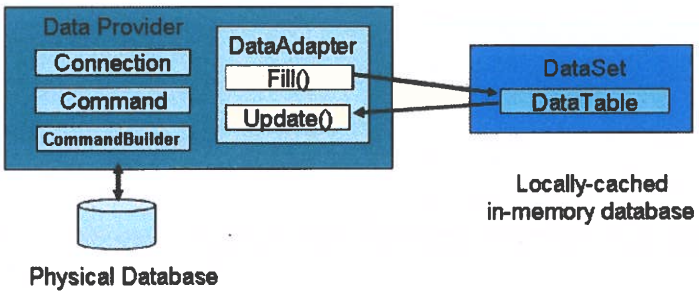

Figura 11. A influência do DataAdapter. ***Fonte:*** *emmet-gray (s.d)*

Vantagens em usar o ADO.NET

A tecnologia criada pela Microsoft, possuí inúmeras vantagens para a sua utilização, os seus principais benefícios são: um sistema desenhado para ambientes "offline" e isso é uma vantagem porque pode-se trabalhar a qualquer altura e pode efectuar-se uma ligação à base de dados apenas quando o utilizador entender que deve ser feita, (Ferreira, 2004) um modelo de programação com suporte avançado para XML (Besteiro & Rodriguez, 2015), um conjunto de classes, interfaces, estruturas e enumerações que gerem o acesso a dados dentro do *framework*. Quando se trabalha em ambientes conectados é mais fácil exercer segurança ao nível do ambiente, a concorrência é mais fácil de controlar e os dados estão sempre mais actualizados do que em outros cenários (como por exemplo quando se trabalha com o ambiente desconectado), (Ferreira, 2004).

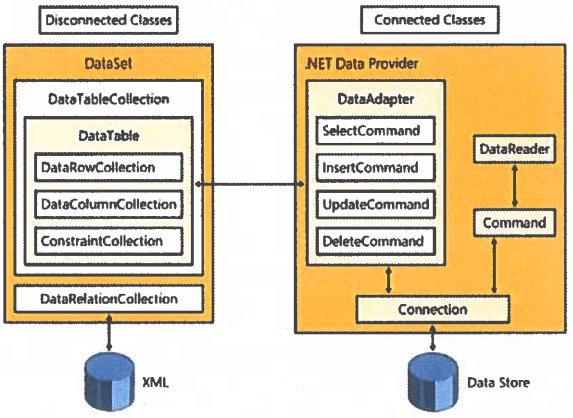

Figura 12. Um comparativo em que se notam as diferenças entre trabalhar no ambiente desconectado e conectado. ***Fonte:*** *Safari Books Online (s.d.)*

II. Contextualização

O pressuposto que desencadeou a razão de ter escolhido este tema deveu-se ao progresso que estas tecnologias têm tido ao longo dos anos e o impacto que têm (e vão continuar a ter no futuro). Os termos de *Cloud Computing* e de Virtualização são falados desde há muitas décadas atrás, mas apenas neste século deu-se um crescimento exponencial sobre estas duas tecnologias e actualmente são muitas as empresas que oferecem serviços de hospedagem na nuvem (Google ou Amazon são alguns exemplos) e também disponibilizam a criação de máquinas virtuais de servidores (como é o caso do Portal Azure, da Microsoft).

Apesar de alguma desconfiança no início devido a uma certa falta de segurança por parte destes serviços, com a chegada de cada vez mais profissionais e especialistas na área, nos dias de hoje, a Virtualização deu um passo em frente e está presente em muitas empresas que possuem um bom conforto monetário para pagar os elevados custos iniciais de uma nuvem.

O SQL tem uma grande importância porque é a principal e a mais utilizada linguagem padrão para gerir e consultar (manipular) uma base de dados. Devido à facilidade de uso, e o facto de ser intuitivo, faz com que o SQL, actualmente ainda continue a ser a referência quando se fala na manipulação de base de dados.

O ADO.NET, criado pela Microsoft, não é mais que um conjunto de classes (a principal tecnologia de acesso de dados para linguagem .NET) que vai interligar (através de código) com o SQL Server para aceder aos dados armazenados numa base de dados remota, além de executar comandos e recuperar resultados.

Se separado estas tecnologias são importantes então quando fazemos uma junção de todas elas – que é exactamente o que vai acontecer na aplicação – aí percebemos realmente que a finalidade é ainda mais importante e o pêndulo necessário para todo este trabalho.

Desenvolvimento

O objectivo do trabalho prático prendia-se com a criação de cinco máquinas virtuais, essas máquinas iriam armazenar a aplicação a ser feita, o servidor de SQL, a partilha de rede e um *Domain Controller* para fazer a ligação entre as máquinas.

A primeira máquina virtual criada é a de *Hyper-V*, porque "dentro" dessa máquina, criaram-se mais quatro máquinas – através da geração 1 do *Hyper-V* – uma de *Domain Controller* (para a criação do domínio), outra *Routing* (para 'espalhar' a rede por todas as máquinas do tipo NAT), uma SQL *Server* (que como o nome indica, armazena o servidor de SQL onde tem a base de dados) e, por última, uma máquina cliente (onde irá correr a aplicação).

O processo seguinte então foi a criação da primeira máquina virtual através do *Hyper-V*, máquina essa chamada de *Domain Controller* em *Windows Server 2012*. Usada para criar o domínio 'pfam.local' com uma rede de IP: 10.1.1.1 e a criação dos *users* que podem usar a aplicação.

A segunda máquina, tal como a primeira em *Windows Server*, a *Routing*, criada com uma placa interna (para comunicar com as máquinas do domínio) e com uma placa de rede externa com o objectivo de, através do *NAT*, conseguir que todas as máquinas que se ligassem ao domínio tivessem conexão Internet. Para isso ser possível, criou-se dois tipos de *Virtual Switch*, o primeiro foi um *Private Virtual Switch* que foi adicionado a todas as máquinas virtuais 'dentro' do *Hyper-V* que tem como finalidade as máquinas poderem comunicar entre si, o segundo é um *External Virtual Switch* que apenas foi adicionado na máquina *Routing*, para, como já foi referido, fazer a comunicação com a máquina física e depois através de *NAT*, espalhar a rede para todas as outras máquinas virtuais.

A última máquina criada com *Windows Server* foi a máquina que irá armazenar tudo relacionado com o SQL – inclusive a base de dados da aplicação. Ou seja, será sempre necessário ter a máquina virtual que alberga o SQL ligada para a conexão com a base de dados seja sempre possível.

Por último, a máquina cliente, esta máquina virtual usa o sistema operativo Windows 8.1, exclusivamente criada para usar a aplicação criada.

Name	nvarchar(50)	☐
Mail	nvarchar(50)	☐
Nationality	nvarchar(50)	☐
Country	nvarchar(50)	☐
City	nvarchar(50)	☐
Address	nvarchar(50)	☑
Gender	nvarchar(50)	☑
Birthday	nvarchar(50)	☐
NIF	nvarchar(50)	☐
Phone	nvarchar(50)	☑
Image	image	☐

Figura 13. *Campos da base de dados criadas no Server Management*

O que realizei primeiro foi a criação da base de dados (fig. 13). Como resolvi criar uma aplicação com uma lista de contactos, criei por isso onze campos que achei mais importante: Nome, E-mail, Nacionalidade, País, Cidade, Morada, Género, Data de Nascimento, NIF, Nº de Telemóvel e, finalmente, o campo Imagem sendo que o campo NIF foi constituído como a nossa chave primária, isto porque não existe mais que uma pessoa com o mesmo número de contribuinte.

A aplicação criada em *Windows Forms* é uma aplicação do estilo *CRUD*, em que é possível criar, ler, editar e eliminar dados que na base de dados. Esta aplicação possuí dois forms: o primeiro com o login, onde será necessário introduzir os dados de um utilizador criado no *Active Directory* (no *DC*) para puder entrar na aplicação e tem um segundo form que é aí que o utilizador pode fazer o *CRUD*.

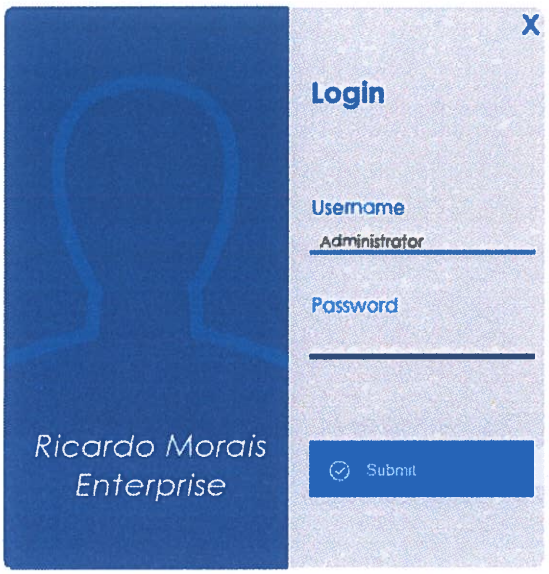

*Figura 14. Interface da aplicação. **Fonte:** Do autor*

A interface da aplicação no login (ver fig. 14) é simples. Possuí as duas caixas de texto - para inserir o nome do utilizador e a password do mesmo - e um botão que levará para outra form caso os dados inseridos existam no Active Directory do Domain Controller.

```
private bool UserAuthentication(String path, String domUsu, String pass)
{
    DirectoryEntry de = new DirectoryEntry(path, domUsu, pass, AuthenticationTypes.Secure);

    try
    {
        DirectorySearcher ds = new DirectorySearcher(de);
        ds.FindOne();
        return true;
    }

    catch
    {
        return false;
    }
}
```

*Figura 15. Serviços de Directório para o Active Directory. **Fonte:** Do autor*

Essa verificação (ver fig. 15) é feita através dos *Directory Services* que são fornecidos pela plataforma .NET. O *DirectoryEntry* representa um item individual num *container* no interior do diretório. De seguida o *DirectorySearcher* que vai executar consultas *com* relação ao *Active Directory* e através do parâmetro *FindOne*, que tem como finalidade "buscar" a primeira entrada do resultado. Tudo inserido dentro de uma classe privada booleana com três *strings* (path, domUso e pass) criadas que serão usadas quando a criação do evento do botão.

O evento (o chamado "*click*") do botão *submit* é onde se irá ter a confirmação se os dados existem ou não no *AD*.

```
private void btnsubmit_Click(object sender, EventArgs e)
{


    string path = @"LDAP://DC1.PFAM.LOCAL";
    string domain = @"pfam.local";
    string user = lblusername.Text.Trim();
    string pass = lblpassword.Text.Trim();
    string domUsu = domain + @"\" + user;

    bool permission = UserAuthentication(path, domUsu, pass);
    if(permission)
    {
        this.Hide();
        CRUD cs = new CRUD();
        cs.ShowDialog();
    }

    else
    {
        lblerror.Visible = true;
    }

}
```

Figura 16. Evento do botão 'submit'. ***Fonte:*** *Do autor*

A criação das strings do evento do botão (ver fig. 16), a *string path* contém o caminho do nosso *LDAP* (que está no *Domain Controller*). A *string domain* é o domínio que criamos aquando a criação da nova floresta no *DC*.

Tanto a *string user* como a *pass* são guardar o conteúdo escrito nas caixas de texto e o método *Trim*() vai remover todos os caracteres de espaço em branco à direita e à esquerda do objecto da *string* actual. Por fim, a última *string* é a que vai juntar a *string* domínio com a string do *username*.

De seguida chamamos a classe booleana que criamos anteriormente e chamamos as respectivas *strings* que também haviam sido criadas e depois temos a condição que se for verdadeiro, a janela vai fechar e abrir-se-á uma nova janela e será aí que é possível introduzir os dados. Se não for verdadeira, retorna uma mensagem de erro que os dados introduzidos estão errados e para tentar novamente.

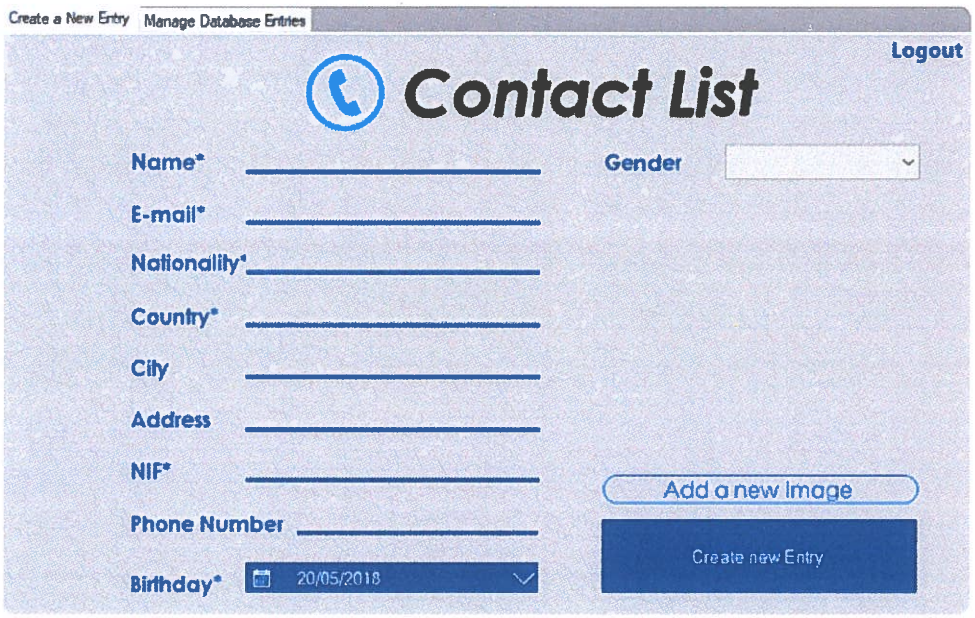

Figura 17. Interface da introdução dos dados. ***Fonte:*** *Do autor*

Na figura 17., é possível ver a parte da introdução dos dados. São no total 11 campos para preencher sendo que 7 deles (os que contém o asterisco e o campo imagem) são de preenchimento obrigatório. Até não preencher esses campos obrigatórios, os dados não serão introduzidos na base de dados.

```
private void btnImage_Click_1(object sender, EventArgs e)
{
    try
    {
        OpenFileDialog dlg = new OpenFileDialog();
        dlg.Filter = "JPG Files (*.jpg)|*.jpg| PNG Files (*.png)|*png|All Files (*.*)|";
        if (dlg.ShowDialog() == DialogResult.OK)
        {
            imgLoc = dlg.FileName.ToString();
            picEmp.ImageLocation = imgLoc;
        }
    }
    catch (Exception ex)
    {
        MessageBox.Show(ex.Message);
    }
}
```

Figura 18. Evento do botão para adicionar a imagem ***Fonte:*** *Do autor*

O botão para inserir a imagem (fig. 18) começa com um *OpenFileDialog* que vai permitir abrir a janela para escolher a imagem pretendida e posteriormente o *Filter* que tem como objectivo apenas permitir ficheiros de imagem do tipo *.png* e *.jpg*. A condição do *if* permite que se o resultado retornar verdadeiro, o campo imgLoc vai ser escrito com o *FileName* e vai mostrar na *PictureBox* (chamada de *picEmp*) a imagem que foi escolhida.

```
private void btnsubmit1_Click_1(object sender, EventArgs e)
{

    if (string.IsNullOrWhiteSpace(nametxt.Text) || string.IsNullOrWhiteSpace(mailtxt.Text) || string.IsNullOrWhiteSpace(birthdaypicker.Text) ||
        string.IsNullOrWhiteSpace(nationalitytxt.Text) || string.IsNullOrWhiteSpace(countrytxt.Text) || string.IsNullOrWhiteSpace(niftxt.Text))
    {
        MessageBox.Show("You need to fill all the required fields.", "Failed to save new entry",
            MessageBoxButtons.OK, MessageBoxIcon.Information);

    }
    else if ((DateTime.Now.Subtract(birthdaypicker.Value).Days) / (365) < 18)
    {
        MessageBox.Show("Less than 18 Years are not allowed.", "Failed to save new entry",
            MessageBoxButtons.OK, MessageBoxIcon.Information);
    }

    else if (Validator.EmailIsValid(mailtxt.Text) == false)
    {
        MessageBox.Show("Your e-mail address is invalid. Please try again.", "Failed to save new entry",
            MessageBoxButtons.OK, MessageBoxIcon.Information);
    }

    else if (IsValidContrib(niftxt.Text) == false)
    {
        MessageBox.Show("Your NIF is invalid. Please try again.", "Failed to save new entry",
            MessageBoxButtons.OK, MessageBoxIcon.Information);
```

Figura 19. Restrições para o campo de inserir dados. ***Fonte:*** *Do autor*

A criação para inserir os dados na base de dados (fig. 19) tem diversas restrições antes de se conseguir adicionar a entrada na base de dados. A primeira restrição é que o campo nome, e-mail, nacionalidade, país, data de nascimento e o NIF não podem estar em branco. Se isso acontecer, então a data de nascimento nunca poderá ser inferior a 18 anos. Se o valor retornar verdadeiro, vai verificar sem o e-mail escrito é valido ou não e se isso acontecer, vai por último, verificar se o NIF introduzido existe ou não. Será tudo explicado nas figuras abaixo apresentadas.

```
public static class Validator
{

    static Regex ValidEmailRegex = CreateValidEmailRegex();

    private static Regex CreateValidEmailRegex()
    {
        string validEmailPattern = @"^(?!\.)(""([^""\r\\]|\\[""\r\\])*""|"
            + @"([-a-z0-9!#$%&'*+/=?^_`{|}~]|(?<!\.)\.)*)(?<!\.)"
            + @"@[a-z0-9][\w\.-]*[a-z0-9]\.[a-z][a-z\.]*[a-z]$";

        return new Regex(validEmailPattern, RegexOptions.IgnoreCase);
    }

    internal static bool EmailIsValid(string emailAddress)
    {
        bool isValid = ValidEmailRegex.IsMatch(emailAddress);

        return isValid;
    }
}
```

Figura 20. Validação do campo e-mail. ***Fonte:*** *Do autor*

A restrição do campo e-mail (fig. 20) pode-se ver que é usado a propriedade Regex (que representam uma expressão regular imutável), essa propriedade é usada para criar então uma classe estática privada onde é declarada uma String em que o campo preenchido terá de obedecer a todos aqueles campos. Posteriormente é criada uma classe interna em que é dito que a variável booleana é igual à string criada anteriormente.

```
public bool IsValidContrib(string nif)
{
    if (!Regex.IsMatch(nif, @"\d{9}"))
    {
        return false;
    }

    char firstChar = nif[0];
    if (firstChar.Equals('1')
        || firstChar.Equals('2')
        || firstChar.Equals('5')
        || firstChar.Equals('6')
        || firstChar.Equals('8')
        || firstChar.Equals('9'))
    {
        int checkDigit = (Convert.ToInt32(firstChar.ToString()) * 9);
        for (int i = 2; i <= 8; ++i)
        {
            checkDigit += Convert.ToInt32(nif[i - 1].ToString()) * (10 - i);
        }

        checkDigit = 11 - (checkDigit % 11);
        if (checkDigit >= 10)
        {
            checkDigit = 0;
        }

        if (checkDigit.ToString() == nif[8].ToString())
        {
            return true;
        }
    }

    return false;
}
```

Figura 21. Restrição do campo NIF, chave primária. ***Fonte:*** *Do autor*

A restrição criada para o NIF (fig. 21) é um pouco mais complexa em relação à anterior. Mais uma vez é usada a propriedade Regex, é usada numa condição if onde terá de ter obrigatoriamente apenas nove números. De seguida criação de uma variável char e ir buscar a primeira posição do NIF e essa posição tinha de começar com os números 1 ou 2 ou 5 ou 6 ou 8 ou 9. Se a primeira posição for algum destes números, então o IF retorna verdadeiro e entra na sua própria condição, então depois é criada uma variável checkDigit (dígito de controlo), o último dígito do NIF é obtido da seguinte forma : 9*d1 + 8*d2 + 7*d3 + 6*d4 + 5*d5 + 4*d6 + 3*d7 + 2*d8 + 1*d9 (em que d1 a d9 são os 9 dígitos do NIF); esta soma terá de ser múltiplo de 11; de seguida, subtrai-se o resto da divisão da soma por 11 a 11 e concluindo, se o resultado for 10, é assumido o algarismo 0;

```
byte[] img = null;
FileStream fs = new FileStream(imgLoc, FileMode.Open, FileAccess.Read);
BinaryReader br = new BinaryReader(fs);
img = br.ReadBytes((int)fs.Length);
string connectionString = "Data Source=DESKTOP-N0B75C5;Initial Catalog=cruddatabase;Integrated Security=True";
SqlConnection sqlConn = new SqlConnection(connectionString);
sqlConn.Open();
```

Figura 22. Método usado para inserir a imagem na base de dados. ***Fonte:*** *Do autor*

Se então todas as condições forem "igual" a falso, a condição do if passará então para o *else* e vai inserir os dados na base de dados. Uma das complicações que tive foi a inserção de

uma imagem na base de dados, após muitas complicações, o "problema" foi ultrapassado. Em primeiro criamos uma variável Byte do tipo *null* (vazia), de seguida um *filestream*, que fornece um '*Stream*' para um arquivo, dando suporte a operações de leitura e gravação síncronas e assíncronas. O imgLoc é como está guardado o caminho da imagem do botão, e depois através do *FileAccess*, vai ler o conteúdo. Com o *BinaryReader*, o programa vai ler os dados com valores binários e de seguida a variável img que foi criada, será utilizada para ler a quantidade de bytes e o tamanho da imagem.

```
SqlCommand cmd = new SqlCommand("INSERT INTO cruddatabase (Name, Mail, Nationality, Country, City, Address, Gender, " +
    "Birthday, NIF, Phone, Image) VALUES ('" + nametxt.Text + "' , '" + mailtxt.Text + "' , '" + nationalitytxt.Text + "' , '"
    + countrytxt.Text + "' , '" + citytxt.Text + "' , '" + addresstxt.Text + "' , '" + generbox.Text + "' , '"
    + birthdaypicker.Value.ToString("dd/MM/yyyy") + "' , '" + niftxt.Text + "' , '" + phonetxt.Text + "', @img)", sqlConn);
cmd.Parameters.Add(new SqlParameter("@img", img));
cmd.ExecuteNonQuery();

sqlConn.Close();
```

Figura 23. Continuação da inserção dos dados. ***Fonte:*** *Do autor*

Os outros campos para inserir na base de dados não tiveram a mesma complicação, pelo que bastou um simples *insert* e escrever o nome das caixas de texto que acabaria por inserir todos os dados na base de dados.

Após serem introduzidos os dados com sucesso, todos os campos serão limpos e aparecerá uma mensagem a confirmar então que a entrada foi efectuada.

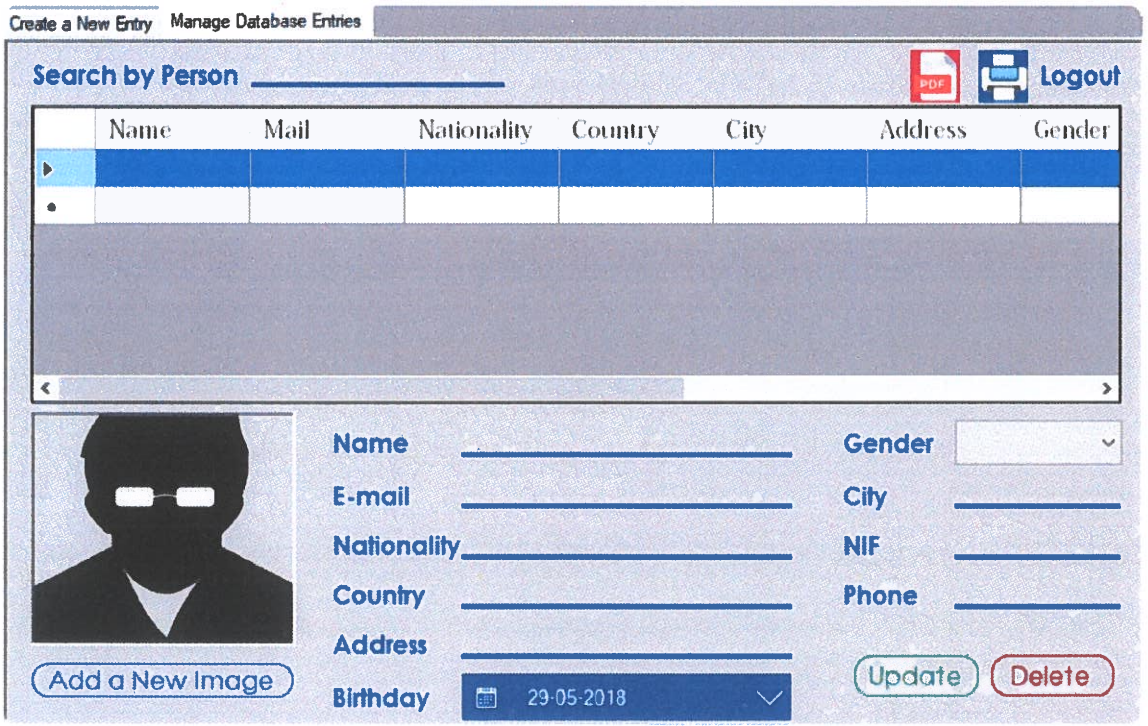

Figura 24. Design do separador Manage Database Entries. ***Fonte:*** *Do autor*

Seguimos então para o separador *Manage Database Entries* (fig. 24) onde o utilizador poderá editar, eliminar ou simplesmente ver os dados que foram introduzidos na base de dados, e ainda pré-visualizar os dados para imprimir ou passar os dados de uma determinada entrada para um *pdf*.

```
private void contactview_CellClick(object sender, DataGridViewCellEventArgs e)
{
    if (e.RowIndex > 0)
    {
        int rowIndex = e.RowIndex;
        DataGridViewRow selectedRow = contactview.Rows[rowIndex];
        textname.Text = selectedRow.Cells[0].Value.ToString();
        textmail.Text = selectedRow.Cells[1].Value.ToString();
        textnat.Text = selectedRow.Cells[2].Value.ToString();
        textcountry.Text = selectedRow.Cells[3].Value.ToString();
        textcity.Text = selectedRow.Cells[4].Value.ToString();
        textaddress.Text = selectedRow.Cells[5].Value.ToString();
        genderbox.Text = selectedRow.Cells[6].Value.ToString();
        textbirthday.Text = selectedRow.Cells[7].Value.ToString();
        textnif.Text = selectedRow.Cells[8].Value.ToString();
        textphone.Text = selectedRow.Cells[9].Value.ToString();

        if (selectedRow.Cells[10].Value != System.DBNull.Value)
        {
            byte[] img = (byte[])(selectedRow.Cells[10].Value);
            MemoryStream ms = new MemoryStream(img);
            pictureBox2.Image = System.Drawing.Image.FromStream(ms);
        }

    }

else
{
    pictureBox2.ImageLocation = "C:/Users/ricar/source/repos/PGForm/PGForm/Resources/Generic person.svg.png";
    pictureBox2.SizeMode = PictureBoxSizeMode.CenterImage;
}
```

Figura 25. O código do Data Grid View que permite visualizar os dados inseridos. ***Fonte:*** *Do autor*

Os dados que foram inseridos são apresentados através de um *Data Grid View* (fig. 25), em que em cada uma das células está apresentado cada campo que existe na base de dados. No campo da imagem, devido a alguns problemas, foi criado uma condição if, para apresentar a imagem introduzida, caso não tenha introduzido qualquer imagem, será apresentada uma imagem por *default* pelo sistema.

```
else
{
    DialogResult dialogResult = MessageBox.Show(String.Format("Do you want do update NIF {0} from the records?", textnif.Text),
        "Update Record", MessageBoxButtons.YesNo);
    if (dialogResult == DialogResult.Yes)
    {
        MessageBox.Show(String.Format("The fields of NIF {0} was been updated!", textnif.Text), "Update Record");
        MemoryStream ms = new MemoryStream();
        pictureBox2.Image.Save(ms, pictureBox2.Image.RawFormat);
        byte[] img = ms.ToArray();
        string connectionString = "Data Source=DESKTOP-N0875C5;Initial Catalog=cruddatabase;Integrated Security=True";
        SqlConnection sqlConn = new SqlConnection(connectionString);
```

Figura 26. Evento do botão editar. ***Fonte:*** *Do autor*

O botão editar, a primeira parte é em tudo igual ao botão de adicionar entrada. Existem todas as condições que são necessárias deixar preenchidas. Se todas essas condições tiverem bem preenchidas, passará então para um else (fig. 26) em que existe a *MessageBox* se o

utilizador quer mesmo editar os dados, se a resposta for sim então passará para o evento de editar os dados na base de dados. Inserir a imagem foi novamente um desafio, o *MemoryStream* cria um fluxo cujo repositório de *backup* é a memória, de seguida a picturebox2 vai salvar a nova imagem introduzida pelo utilizador.

```
SqlDataAdapter da = new SqlDataAdapter();
da.UpdateCommand = new SqlCommand("UPDATE cruddatabase SET Name=@Name, Mail=@Mail, Nationality=@Nationality, Country=@Country, " +
    "City=@City, Address=@Address, Gender=@Gender, Birthday=@Birthday, NIF=@NIF, Phone=@Phone, Image=@img WHERE NIF=@NIF", sqlConn);
da.UpdateCommand.Parameters.Add("@Name", SqlDbType.NVarChar).Value = textname.Text;
da.UpdateCommand.Parameters.Add("@Mail", SqlDbType.NVarChar).Value = textmail.Text;
da.UpdateCommand.Parameters.Add("@Nationality", SqlDbType.NVarChar).Value = textnat.Text;
da.UpdateCommand.Parameters.Add("@Country", SqlDbType.NVarChar).Value = textcountry.Text;
da.UpdateCommand.Parameters.Add("@City", SqlDbType.NVarChar).Value = textcity.Text;
da.UpdateCommand.Parameters.Add("@Address", SqlDbType.NVarChar).Value = textaddress.Text;
da.UpdateCommand.Parameters.Add("@Gender", SqlDbType.NVarChar).Value = genderbox.Text;
da.UpdateCommand.Parameters.Add("@Birthday", SqlDbType.NVarChar).Value = textbirthday.Value;
da.UpdateCommand.Parameters.Add("@NIF", SqlDbType.NVarChar).Value = textnif.Text;
da.UpdateCommand.Parameters.Add("@Phone", SqlDbType.NVarChar).Value = textphone.Text;
da.UpdateCommand.Parameters.Add("@img", SqlDbType.Image).Value = img;

sqlConn.Open();
da.UpdateCommand.ExecuteNonQuery();
sqlConn.Close();
```

Figura 27. Continuação do evento editar. ***Fonte:*** *Do autor*

A continuação do evento editar (fig. 27) é também ela muito parecida ao evento de inserir, a principal diferença é que no comando de SQL (a que se deu o nome *da.UpdateCommand*) usamos o *update* que é usado para editar dados. De seguida, esse *da.UpdateCommand* chamada o parâmetro *add* que editará os campos pretendidos na base de dados. Apesar de estarem todos os campos, obviamente o utilizador não necessita de preencher todos os campos para o código funcionar, apenas é fulcral preencher os campos que são obrigatórios aquando a criação da entrada que havia sido feita.

```
private void btndelete_Click_1(object sender, EventArgs e)
{
    string connectionString = "Data Source=DESKTOP-N0B75C5;Initial Catalog=cruddatabase;Integrated Security=True";
    SqlConnection sqlConn = new SqlConnection(connectionString);
    sqlConn.Open();
    SqlCommand cmd = sqlConn.CreateCommand();
    cmd.CommandType = CommandType.Text;
    cmd.CommandText = "DELETE FROM cruddatabase WHERE NIF='" + this.textnif.Text + "'";
    cmd.ExecuteNonQuery();
    sqlConn.Close();
    disp_data();
}
```

Figura 28. Evento eliminar quando é clicado. ***Fonte:*** *Do autor*

Se o que o utilizador pretende é eliminar os dados, o código é bem menos difícil que os dois anteriores (fig. 28), a conexão é feita, abre-se a mesma, criou-se um comando próprio de SQL chamado de *cmd*, de seguida diz-se que esse comando é de texto e então depois o *cmd.CommandText* tem a frase em SQL para apagar na base de dados onde o NIF será igual à *textbox* que contém o NIF (nossa chave primária e, por isso, não existe mais ninguém com o

mesmo NIF). De seguida, o *ExecuteNonQuery* fará então o pretendido, a conexão é fechada e temos o *disp_data()* que ainda não falei dele (irei falar no parágrafo a seguir) mas que serve para actualizar os campos do *Data Grid View*.

```
public void disp_data()
{
    string connectionString = "Data Source=DESKTOP-N0B75C5;Initial Catalog=cruddatabase;Integrated Security=True";
    SqlConnection sqlConn = new SqlConnection(connectionString);
    SqlCommand cmd = sqlConn.CreateCommand();
    sqlConn.Open();
    cmd.CommandType = CommandType.Text;
    cmd.CommandText = "Select * from cruddatabase";
    cmd.ExecuteNonQuery();
    DataTable dt = new DataTable();
    SqlDataAdapter da = new SqlDataAdapter(cmd);
    da.Fill(dt);
    contactview.DataSource = dt;
    sqlConn.Close();
}
```

Figura 29. *A função disp_data que actualiza o Data Grid.* ***Fonte:*** *Do autor*

Temos então o disp_data() (fig. 29) que actualiza sempre o *Data Grid View* mal exista uma alteração (seja uma nova entrada, editar ou apagar) na base de dados. Fará então a conexão à base de dados, abrir-se-á essa conexão, criação de um comando cmd de texto e onde se fará um *Select* da base de dados (o *Select* serve para mostrar tudo o que existe na base de dados). O *DataTable* representará uma tabela de dados na memória e de seguida um *DataAdapter* que actualiza a base de dados (vai receber a informação contida do comando cmd, ou seja, o *Select* da base de dados) e, em seguida, esse *DataAdapter* será preenchido pelo *DataTable*. O *contactview* (nome da nossa *Data Grid View*) terá como fonte, o que estiver contido naquela DataTable e, portanto, actualizará sempre que houver alguma alteração.

```
private void bunifuMaterialTextbox1_OnValueChanged(object sender, EventArgs e)
{
    string keyword = procurarnome.Text;
    string connectionString = "Data Source=DESKTOP-N0B75C5;Initial Catalog=cruddatabase;Integrated Security=True";
    SqlConnection sqlConn = new SqlConnection(connectionString);
    sqlConn.Open();
    sda = new SqlDataAdapter("SELECT * FROM cruddatabase WHERE Name LIKE '%" + keyword + "%'", sqlConn);
    DataTable dt = new DataTable();
    sda.Fill(dt);
    contactview.DataSource = dt;
}
```

Figura 30. *A função para procurar pelo nome na base de dados.* ***Fonte:*** *Do autor*

Para facilitar a pesquisa caso existam muitas entradas na base de dados, existe a possibilidade nesta aplicação de o utilizador pesquisar pelo nome, assim, ficará mais fácil para encontrar determinada pessoa. Para isso, criou-se um evento que quando a caixa de texto tem

o seu valor alterado, ou seja, alguém escreveu, essa mesma *textbox* acionará o evento da figura acima (fig. 30). Criou-se uma variável do tipo *string* - *keyword* - a que se deu o valor dessa mesma caixa de texto. De seguida, a conexão à base de dados e ligar essa mesma conexão. Posteriormente, um *SqlDataAdapter* com um comando *Select*, onde vai pesquisar todos os nomes que estejam dentro daquela *string keyword*, por exemplo, basta o utilizador pôr a letra 'r' que o campo irá filtrar por todos os nomes que contenham a letra 'r'. E vai actualizar porque, mais uma vez, vai-se preencher o *Data Grid View* através da função *DataSource*.

```
private void btnprint_Click(object sender, EventArgs e)
{
    if (printPreviewDialog1.ShowDialog() == DialogResult.OK)
    {
        printDocument1.Print();
    }
}

private void printDocument1_PrintPage(object sender, System.Drawing.Printing.PrintPageEventArgs e)
{
    e.Graphics.DrawString(textname.Text, new System.Drawing.Font("Time New Roman", 14, FontStyle.Bold), Brushes.Black, new Point(100, 120));
    e.Graphics.DrawString(textnif.Text, new System.Drawing.Font("Time New Roman", 14, FontStyle.Bold), Brushes.Black, new Point(100, 150));
    e.Graphics.DrawString(textmail.Text, new System.Drawing.Font("Time New Roman", 14, FontStyle.Bold), Brushes.Black, new Point(100, 180));
    e.Graphics.DrawString(textnat.Text, new System.Drawing.Font("Time New Roman", 14, FontStyle.Bold), Brushes.Black, new Point(100, 210));
    e.Graphics.DrawString(textcountry.Text, new System.Drawing.Font("Time New Roman", 14, FontStyle.Bold), Brushes.Black, new Point(100, 240));
    e.Graphics.DrawString(textcity.Text, new System.Drawing.Font("Time New Roman", 14, FontStyle.Bold), Brushes.Black, new Point(100, 270));
    e.Graphics.DrawString(textaddress.Text, new System.Drawing.Font("Time New Roman", 14, FontStyle.Bold), Brushes.Black, new Point(100, 300));
    e.Graphics.DrawString(genderbox.Text, new System.Drawing.Font("Time New Roman", 14, FontStyle.Bold), Brushes.Black, new Point(100, 330));
    e.Graphics.DrawString(textbirthday.Text, new System.Drawing.Font("Time New Roman", 14, FontStyle.Bold), Brushes.Black, new Point(100, 360));
    e.Graphics.DrawString(textphone.Text, new System.Drawing.Font("Time New Roman", 14, FontStyle.Bold), Brushes.Black, new Point(100, 390));
    e.Graphics.DrawImage(pictureBox2.Image, new Point(320, 120));
}
```

Figura 31. *Código para impressão da informação.* ***Fonte:*** *Do autor*

Se o objectivo for imprimir de imediato a informação de alguma entrada na base de dados, o utilizador ao clicar no botão de fundo azul com uma impressora, será levado para uma janela de pré-visualização onde poderá imprimir a informação pretendida. O código em si (fig. 31) é bastante simples. O evento do *click* do botão tem apenas a condição *if* que se caso o resultado seja positivo, chama a função *printDocument1*, essa função vai então mostrar todo o conteúdo das caixas de texto. Serão apresentadas com o tipo de letra *Time New Roman*, com um tamanho de 14 e o seu estilo será em negrito. O que vem a seguir são os *Brushes*, que é a área pintada, neste caso é preta e o *new Point* corresponde ao posicionamento de cada caixa de texto na folha A4 que será imprimida. Na imagem abaixo apresentada (fig. 32) pode-se ver o que acontece quando o utilizador clica na impressora.

Figura 32. *Pré-Visualização do botão print*

```
private void btnpdf_Click(object sender, EventArgs e)
{

    using (SaveFileDialog saveFile = new SaveFileDialog()

    {
        Filter = "PDF file|*.pdf",
        ValidateNames = true
    })
```

Figura 33. Evento após clicar na imagem 'pdf'. ***Fonte:*** *Do autor*

Se a opção em vez de imprimir for apenas extrair os campos da base de dados para um *pdf*, o utilizador ao clicar na imagem em que está escrito *pdf* (fig. 33) abrir-se-á um *OpenFileDialog* que permitirá ao utilizador escolher onde guardar o pdf e em que nome pretende que seja guardado. O *filter* diz-nos que o ficheiro será guardado no formato pdf, o *ValidateNames* obtém (ou define) um valor indicando se o *File Dialog* aceita apenas nomes de arquivos *Win32* válidos.

```
if (saveFile.ShowDialog() == DialogResult.OK)
{
    iTextSharp.text.Document doc = new iTextSharp.text.Document(PageSize.A5.Rotate());
    try
    {
        PdfWriter.GetInstance(doc, new FileStream(saveFile.FileName, FileMode.Create));
        doc.Open();
        doc.Add(new iTextSharp.text.Paragraph(textname.Text));
        doc.Add(new iTextSharp.text.Paragraph(textnif.Text));
        doc.Add(new iTextSharp.text.Paragraph(textmail.Text));
        doc.Add(new iTextSharp.text.Paragraph(textnat.Text));
        doc.Add(new iTextSharp.text.Paragraph(textcountry.Text));
        doc.Add(new iTextSharp.text.Paragraph(textcity.Text));
        doc.Add(new iTextSharp.text.Paragraph(textaddress.Text));
        doc.Add(new iTextSharp.text.Paragraph(genderbox.Text));
        doc.Add(new iTextSharp.text.Paragraph(textphone.Text));
        doc.Add(new iTextSharp.text.Paragraph(textbirthday.Text));
        iTextSharp.text.Image image = iTextSharp.text.Image.GetInstance(pictureBox2.Image, System.Drawing.Imaging.ImageFormat.Jpeg);
        image.ScaleAbsolute(159f, 159f);
        image.SetAbsolutePosition(255, 225);
        doc.Add(image);
    }
```

Figura 34. Continuação do código do botão pdf. ***Fonte:*** *Do autor*

A condição if diz-nos que se tudo correr como suposto (o resultado do *dialog* for verdadeiro), então tudo o que está dentro da condição if começará a funcionar. Foi usado a estrutura de *ITextSharp* porque facilita o processo de criação de *PDF*. Portanto criamos um documento de texto (chamado de *doc*) em que essa página será de tamanho folha A5. Posteriormente é criada uma nova instância e adicionado ao pdf todos os campos de texto (os que não tiverem preenchidos, obviamente ficaram em branco).

Mais uma vez, a imagem deu algum trabalho para aparecer, mas depois de algumas tentativas bastou ir buscar a instância da imagem, identificar a PictureBox onde a imagem estava inserida e depois através do *System.Drawing.Imaging*, demos o formato à imagem e o tipo da mesma, neste caso será *jpeg*. Para efeitos demonstrativos, ao clicarmos então no botão de pdf vai abrir uma janela a perguntar onde queremos guardar o ficheiro (fig. 35)

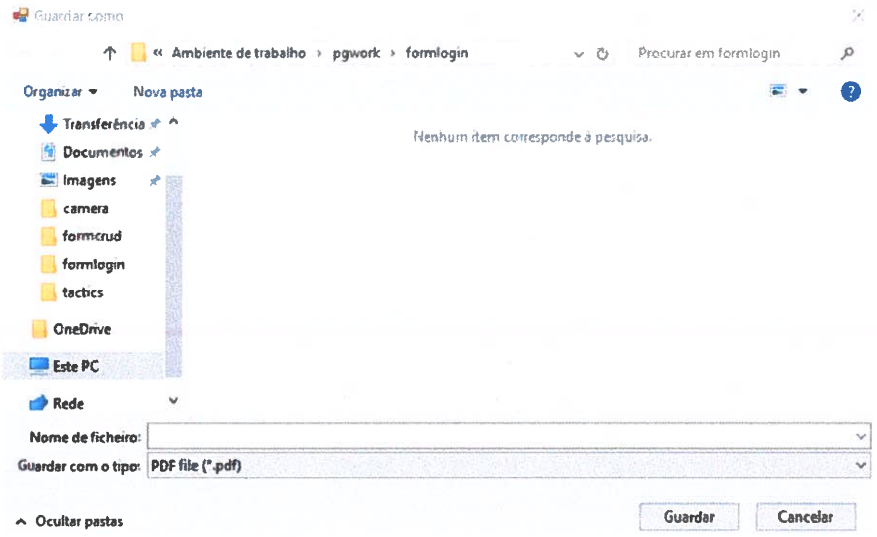

Figura 35. *O "Guardar Como" aparece logo a seguir a clicar no botão de PDF.* ***Fonte:*** *Do autor*

Neste caso vamos dar o nome de "RicardoMoraisEntry.pdf" e guardar numa pasta à escolha. O processo de guardar, nas máquinas virtuais, não é instantâneo e poderá demorar alguns segundos. Após estar concluído, aparecerá o ficheiro *pdf* na pasta que escolhemos para o devido efeito. Ao abrir a disposição do pdf é o que se vê na imagem abaixo (fig. 36), simples, mas prático.

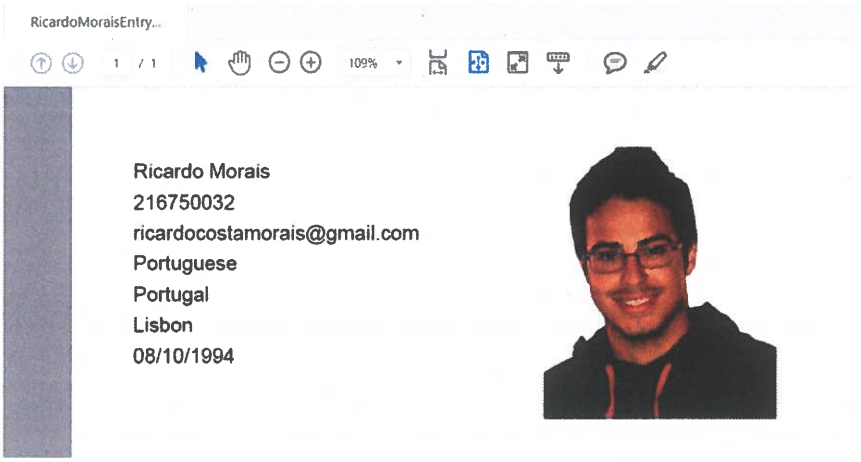

Figura 36. *O conteúdo do pdf e como é apresentado.* ***Fonte:*** *Do autor*

IV. Discussão

Se antigamente a virtualização e a computação na nuvem eram vistas com algumas desconfianças, nos dias de hoje, essa desconfiança tem desvanecendo a um ritmo gradual. Existem inúmeras vantagens ao usar estas tecnologias, mas mesmo assim ainda existe algum desconhecimento por parte das empresas ou simplesmente por falta de familiarização com as mesmas essas tecnologias não são usadas na plenitude dos seus recursos. Mesmo que esse paradigma tenha estado a mudar ainda há um grande caminho a percorrer para tornar tanto a virtualização como o *cloud computing*, olhando para o ponto de vista do cliente e não do fornecedor, algo completamente fundamental que só traria benefícios para a empresa a médio/longo prazo.

O *cloud computing* teve um grande impacto. Desde a sua existência até ao real aparecimento de uma *Cloud* ainda neste século, levou a Internet para um outro nível. Para quê ter tantos discos com imensa capacidade de memória se simplesmente é possível guardar toda essa informação numa *Cloud*? O único custo real é apenas no início, quando é necessário criar as infraestruturas necessárias, depois não é necessário fazer a manutenção normal que se fazia ou alguma avaria que pudesse haver caso existisse algum problema com o(s) disco(s). Só aqui estão duas vantagens que são fulcrais para o cliente implementar uma nuvem. Portanto está mais que patente nos especialistas da área que a computação na nuvem é o presente e o futuro e as empresas mais cedo ou mais tarde vão acabar por ver essas vantagens e vão ser recomendadas a apostar num serviço *cloud*.

Como se pode ver, também a virtualização oferece bastantes vantagens e benefícios principalmente ao nível de segurança (porque cada vez mais existem especialistas na área), redução de custos e disponibilidade. Puder ter um computador com as máquinas virtuais a correrem em vários sistemas operativos é um ajuda tremenda para as empresas, e as próprias empresas nos últimos anos têm-se apercebido disso. A virtualização é, actualmente, vista como um "bem" necessário para qualquer empresa, mesmo com todos os custos inerentes no início acaba por ser altamente compensativo se pensada a um longo prazo.

Conclusão

A tecnologia está em constante crescimento e alteração nos dias de hoje. Se o termo computação na nuvem era praticamente desconhecida para as pessoas no início do século XXI, o que é certo é que actualmente todas as pessoas já ouviram falar do termo nuvem, até mesmo as pessoas que não têm um contacto directo com o meio. O termo nuvem está em constante expansão e são raras as pessoas que não ouviram falar e que nunca usaram nenhum tipo de *cloud*. A *cloud* está cada vez mais implementada na sociedade, quer seja através de forma pessoal (guardar fotos, ficheiros) ou a nível de empresa (guardar grandes ficheiros, de grandes dimensões, por exemplo).

Também a virtualização, como se pode ver, é cada vez mais uma realidade no passado recente. Mesmo com algumas desconfianças naturais no início do seu aparecimento, o que é certo é que cada vez mais empresas adoptam este mesmo sistema para não terem elevados custos a longos prazos e o meu trabalho realizado prova isso mesmo: criei facilmente quatro máquinas virtuais, todas dentro do mesmo domínio a comunicarem-se entre si e com uma aplicação que apenas a máquina cliente poderia usar. Se para o meu trabalho, a virtualização foi fulcral, para uma empresa com milhares de computadores em que se pretende tudo ligado ao mesmo domínio, ainda mais importante esse conteúdo será.

Referências Bibliográficas

Alecrim, E. (25 de Julho de 2012). *O que é a Virtualização e para que serve?* Obtido de InfoWester: https://www.infowester.com/virtualizacao.php Consultado em: 10 de Maio de 2018

Azure, M. (s.d.). *Microsoft*. Obtido de Microsoft Azure: https://azure.microsoft.com/pt-pt/overview/what-is-hybrid-cloud-computing/ Consultado em: 8 de Maio de 2018

BD, C. (2013). *DQL – Linguagem de consulta de dados*. Obtido de DQL – Linguagem de consulta de dados: https://consultabd.wordpress.com/2013/08/15/dql-linguagem-de-consulta-de-dados/ Consultado em: 14 de Maio de 2018

Besteiro, M., & Rodriguez, M. (2015). *ADO.NET.* Consultado em: 17 de Maio

Böhm, M., & Krcmar, H. (21). Cloud Computing and Cloud Evolution. *Research Gate*. Consultado em: 10 de Maio de 2018

Brinda, M., & Heric, M. (25 de Janeiro de 2017). *The Changing Faces of the Cloud.* Obtido de Bain & Company: http://www.bain.com/publications/articles/the-changing-faces-of-the-cloud.aspx Consultado em: 10 de Maio de 2018

Chamberlin, D. D. (2012). *ieeexplore.* Obtido de ieeexplore: https://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=6359709 Consultado em: 10 de Maio de 2018

Chapple, M. (2017). *Data Control Language*. Obtido de Data Control Language: https://www.lifewire.com/data-control-language-dcl-1019477 Consultado em: 14 de Maio de 2018

Cloud21. (Novembro de 2016). *TI Verde: empresas mais sustentáveis com Cloud Computing*. Obtido de Cloud21: https://cloud21.com.br/computacao-em-nuvem/ti-verde-empresas-mais-sustentaveis-com-cloud-computing/ Consultado em: 11 de Maio de 2018

ComputerWorld. (2018). *Computer World*. Obtido de 3 produtos inovadores na segurança de redes: https://www.computerworld.com.pt/2018/01/03/3-produtos-inovadores-na-seguranca-de-redes/ Consultado em: 12 de Maio de 2018

Costa, R. F. (2009). *INTEL VIRTUALIZATION TECHNOLOGY*. Obtido de INTEL VIRTUALIZATION TECHNOLOGY:

https://www.gta.ufrj.br/ensino/eel879/trabalhos_vf_2009_2/romulo/Intel_Virtualization_Technologies/Hypervisor.html Consultado em: 13 de Maio de 2018

Dybka, P. (2014). *Vertabelo*. Obtido de Vertabelo: http://www.vertabelo.com/blog/notes-from-the-lab/sql-or-sequel Consultado em: 15 de Maio de 2018

Endeavor. (2015). *Hoje, qualquer empresa precisa de um banco de dados. E qualquer banco de dados precisa de SQL.* Obtido de Hoje, qualquer empresa precisa de um banco de dados. E qualquer banco de dados precisa de SQL.: https://endeavor.org.br/sql/ Consultado em: 14 de Maio de 2018

Espojeira, I. (Setembro de 2013). *SaaS: o que é e para que me serve?* Obtido de https://invoicexpress.com/blog/saas: https://invoicexpress.com/blog/saas Consultado em: 12 de Maio de 2018

Fernando. (2014). *Blue Solutions*. Obtido de Os 12 Benefícios da Virtualização de Servidores no Datacenter: https://www.bluesolutions.com.br/2014/07/os-12-beneficios-da-virtualizacao-de-servidores-no-datacenter/ Consultado em: 12 de Maio de 2018

Ferreira, N. (2004). *Uma Visão Geral do ADO.NET.* Porto. Consultado em: 16 de Maio

Google. (2018). *Linguagem de definição de dados*. Obtido de: https://cloud.google.com/spanner/docs/data-definition-language?hl=pt-br Consultado em: 16 de Maio de 2018

Google. (s.d.). Obtido de https://cloud.google.com/bigquery/docs/reference/standard-sql/data-manipulation-language Consultado em: 13 de Maio de 2018

GrupoCloud. (s.d.). *Cloud Computing e sustentabilidade, é possível?* Obtido de Group Cloud: https://grupocloud.com.br/2018/02/14/cloud-computing-e-sustentabilidade-e-possivel/ Consultado em: 9 de Maio de 2018

Halvorsen, H.-P. (2016). *Structured Query Language.* Obtido de http://home.hit.no/~hansha/documents/database/documents/Structured%20Query%20Language.pdf Consultado em: 13 de Maio de 2018

Húngaro, L. (2015). *Uma Visão Geral do ADO.NET.* Obtido de Uma Visão Geral do ADO.NET: http://www.linhadecodigo.com.br/artigo/435/uma-visao-geral-do-adonet.aspx Consultado em: 15 de Maio de 2018

InforTrend. (s.d.). *InforTrendBrasil*. Obtido de Virtualização - O que é virtualização?: http://www.infortrendbrasil.com.br/virtualizacao/ Consultado em: 12 de Maio de 2018

IPM. (1 de Setembro de 2017). *História da computação em nuvem: como surgiu a cloud computing?* Obtido de ipm: https://www.ipm.com.br/blog/historia-da-computacao-em-nuvem-como-surgiu-a-cloud-computing/ Consultado em: 10 de Maio de 2018

Kaupa, P. (2014). *ADO.Net - Fundamentos*. Obtido de ADO.Net - Fundamentos: https://www.devmedia.com.br/ado-net-fundamentos/9342 Consultado em: 15 de Maio de 2018

KerridgeCS. (s.d.). *Kerridgecs.* Obtido de https://cdn2.hubspot.net/hubfs/620276/KCS-UK/KCS-UK%20Whitepapers/History_of_Cloud_Computing_Timeline.pdf Consultado em: 11 de Maio de 2018

LearningSolutions. (2017). *NH Learning Solutions*. Obtido de NH Learning Solutions: https://blog.nhlearningsolutions.com/blog/4-key-benefits-of-desktop-virtualization Consultado em: 12 de Maio de 2018

Monginho, M. (2012). *Estudo do impacto da virtualização de hardware.* Setúbal. Consultado em: 12 de Maio de 2018

Okano, M. (2008). *O impacto da virtualização nas empresas.* Obtido de Research Gate: https://www.researchgate.net/publication/301544214_O_impacto_da_virtualizacao_nas_empresas Consultado em: 12 de Maio de 2018

Pinheiro, F. (Fevereiro de 2010). *GTA/URFJ - Universidade Federal do Rio de Janeiro*. Obtido de Cloud Computing: https://www.gta.ufrj.br/ensino/eel879/trabalhos_vf_2010_2/fernando/caracteristicas.html Consultado em: 9 de Maio de 2018

Reilly, D. (2005). *Should you use ADO.NET DataReader or DataSet?* Obtido de Should you use ADO.NET DataReader or DataSet?: https://www.red-gate.com/simple-talk/dotnet/net-framework/should-you-use-ado-net-datareader-or-dataset/ Consultado em: 16 de Maio de 2018

Rocha, V. (2013). *TI Especialistas*. Obtido de TI Especialistas: https://www.tiespecialistas.com.br/tipos-de-virtualizacao/ Consultado em: 11 de Maio

Rosa, A. (2013). *Windows Server 2012.* Lisboa: FCA. Consultado em: 8 de Maio de 2018

Sousa, F. R., Machado, J., & Moreira, L. (2011). *ResearchGate.* Obtido de ResearchGate: https://www.researchgate.net/profile/Javam_Machado/publication/237644729_Computacao_em_Nuvem_Conceitos_Tecnologias_Aplicacoes_e_Desafios/links/56044f308aea25fce3121f3/Computacao-em-Nuvem-Conceitos-Tecnologias-Aplicacoes-e-Desafios.pdf Consultado em: 11 de Maio de 2018

Thakur, S. (2017). *whatisdbms*. Obtido de whatisdbms: http://whatisdbms.com/what-is-sql-applications-advantages-and-disadvantages/ Consultado em: 10 de Maio de 2018

VirtueIT. (2015). *VirtueIT*. Obtido de VirtueIT: http://www.virtueit.com.br/site/?page_id=463 Consultado em: 10 de Maio de 2018

VMware. (s.d.). Obtido de https://www.vmware.com/br/solutions/virtualization.html Consultado em: 12 de Maio de 2018

Zaïane, O. R. (2016). Obtido de https://www.cs.sfu.ca/CourseCentral/354/zaiane/material/notes/Chapter1/node17.html : Consultado em 14 de Maio de 2018

Zanutto, B. G. (s.d.). *Segurança em Cloud Computing* . Consultado em 14 de Maio de 2018

Anexos

Anexo 1 – Código do Funcionamento do Login

```
1.  using System;
2.  using System.Collections.Generic;
3.  using System.ComponentModel;
4.  using System.Data;
5.  using System.Drawing;
6.  using System.Linq;
7.  using System.Text;
8.  using System.Threading.Tasks;
9.  using System.Windows.Forms;
10. using System.DirectoryServices;
11.
12. namespace PGForm
13. {
14.     public partial class MainForm : Form
15.     {
16.         public MainForm()
17.         {
18.             InitializeComponent();
19.
20.         }
21.
22.         private void Form1_Load(object sender, EventArgs e)
23.         {
24.
25.         }
26.
27.         private void btnsubmit_Click(object sender, EventArgs e)
28.         {
29.
30.
31.             string path = @"LDAP://DC1.PFAM.LOCAL";
32.             string domain = @"pfam.local";
33.             string user = lblusername.Text.Trim();
34.             string pass = lblpassword.Text.Trim();
35.             string domUsu = domain + @"\" + user;
36.
37.             bool permission = UserAuthentication(path, domUsu, pass);
38.             if (permission)
39.             {
40.                 this.Hide();
41.                 CRUD cs = new CRUD();
42.                 cs.ShowDialog();
43.             }
44.
45.             else
46.             {
47.                 lblerror.Visible = true;
48.             }
```

```
49.
50.        }
51.
52.        private bool UserAuthentication(String path, String domUsu, String pass)
53.        {
54.            DirectoryEntry de = new DirectoryEntry(path, domUsu, pass, AuthenticationT
    ypes.Secure);
55.
56.            try
57.            {
58.                DirectorySearcher ds = new DirectorySearcher(de);
59.                ds.FindOne();
60.           return true;
61.            }
62.
63.            catch
64.            {
65.                return false;
66.            }
67.        }
68.
69.        private void lblexit_Click(object sender, EventArgs e)
70.        {
71.            Application.Exit();
72.        }
73.
74.    }
75. }
```

Anexo 2 – Código do Funcionamento do CRUD

```
1.  using System;
2.  using System.Data;
3.  using System.IO;
4.  using System.Drawing;
5.  using System.Windows.Forms;
6.  using System.Data.SqlClient;
7.  using iTextSharp.text;
8.  using iTextSharp.text.pdf;
9.  using System.Text.RegularExpressions;
10.
11. namespace PGForm
12. {
13.     public partial class CRUD : Form
14.     {
15.
16.         SqlDataAdapter sda;
17.         String imgLoc = "";
```

```
18.
19.     public CRUD()
20.     {
21.         InitializeComponent();
22.         this.genderbox.DropDownStyle = ComboBoxStyle.DropDownList;
23.     }
24.     private void btn2_Click_1(object sender, EventArgs e)
25.     {
26.         try
27.         {
28.             OpenFileDialog dlg = new OpenFileDialog();
29.             dlg.Filter = "JPG Files (*.jpg)|*.jpg| PNG Files (*.png)|*png|All Files (*.*)|
    *.*";
30.             if (dlg.ShowDialog() == DialogResult.OK)
31.             {
32.                 imgLoc = dlg.FileName.ToString();
33.                 pictureBox2.ImageLocation = imgLoc;
34.             }
35.         }
36.         catch (Exception ex)
37.         {
38.             MessageBox.Show(ex.Message);
39.         }
40.     }
41.
42.     private void btnimage_Click_1(object sender, EventArgs e)
43.     {
44.         try
45.         {
46.             OpenFileDialog dlg = new OpenFileDialog();
47.             dlg.Filter = "JPG Files (*.jpg)|*.jpg| PNG Files (*.png)|*png|All Files (*.*)|
    *.*";
48.             if (dlg.ShowDialog() == DialogResult.OK)
49.             {
50.                 imgLoc = dlg.FileName.ToString();
51.                 picEmp.ImageLocation = imgLoc;
52.             }
53.         }
54.         catch (Exception ex)
55.         {
56.             MessageBox.Show(ex.Message);
57.         }
58.     }
59.
60.
61.     private void lblout_Click(object sender, EventArgs e)
62.     {
63.         this.Hide();
64.         MainForm voltar = new MainForm();
65.         voltar.ShowDialog();
```

```
66.         }
67.
68.
69.         private void lblsair_Click(object sender, EventArgs e)
70.         {
71.             this.Hide();
72.             MainForm voltar = new MainForm();
73.             voltar.ShowDialog();
74.         }
75.
76.         private void btnupdate_Click_1(object sender, EventArgs e)
77.         {
78.             if (string.IsNullOrWhiteSpace(textname.Text) || string.IsNullOrWhiteSpace(t
    extmail.Text) || string.IsNullOrWhiteSpace(textnationality.Text)
79.                 || string.IsNullOrWhiteSpace(textcountry.Text) || string.IsNullOrWhiteSpac
    e(textnif.Text))
80.             {
81.                 MessageBox.Show("You need to fill all the required fields.", "Failed to sav
    e new entry", MessageBoxButtons.OK, MessageBoxIcon.Information);
82.
83.             }
84.             else if ((DateTime.Now.Subtract(textbirthday.Value).Days) / (365) < 18)
85.             {
86.                 MessageBox.Show("Less than 18 Years are not allowed", "Failed to save ne
    w entry", MessageBoxButtons.OK, MessageBoxIcon.Information);
87.             }
88.
89.             else if (Validator.EmailIsValid(textmail.Text) == false)
90.             {
91.                 MessageBox.Show("Your e-
    mail address is invalid. Please try again.", "Failed to save new entry", MessageBoxBu
    ttons.OK, MessageBoxIcon.Information);
92.             }
93.
94.             else if (IsValidContrib(textnif.Text) == false)
95.             {
96.                 MessageBox.Show("Your NIF is invalid. Please try again.", "Failed to save
    new entry", MessageBoxButtons.OK, MessageBoxIcon.Information);
97.             }
98.
99.             else
100.                {
101.                    DialogResult dialogResult = MessageBox.Show(String.Format("Do
    you want do update NIF {0} from the records?", textnif.Text), "Update Record", Mes
    sageBoxButtons.YesNo);
102.                    if (dialogResult == DialogResult.Yes)
103.                    {
104.                        MessageBox.Show(String.Format("The fields of NIF {0} was bee
    n updated!", textnif.Text), "Update Record");
105.                        MemoryStream ms = new MemoryStream();
```

```
106.                    pictureBox2.Image.Save(ms, pictureBox2.Image.RawFormat);
107.                    byte[] img = ms.ToArray();
108.                    string connectionString = "Data Source=DESKTOP-
    N0B75C5;Initial Catalog=cruddatabase;Integrated Security=True";
109.                    SqlConnection sqlConn = new SqlConnection(connectionString);

110.
111.                    SqlDataAdapter da = new SqlDataAdapter();
112.                    da.UpdateCommand = new SqlCommand("UPDATE cruddatabas
    e SET Name=@Name, Mail=@Mail, Nationality=@Nationality, Country=@Country,
     " +
113.                        "City=@City, Address=@Address, Gender=@Gender, Birthda
    y=@Birthday, NIF=@NIF, Phone=@Phone, Image=@img WHERE NIF=@NIF", sql
    Conn);
114.                    da.UpdateCommand.Parameters.Add("@Name", SqlDbType.NVa
    rChar).Value = textname.Text;
115.                    da.UpdateCommand.Parameters.Add("@Mail", SqlDbType.NVar
    Char).Value = textmail.Text;
116.                    da.UpdateCommand.Parameters.Add("@Nationality", SqlDbType
    .NVarChar).Value = textnat.Text;
117.                    da.UpdateCommand.Parameters.Add("@Country", SqlDbType.N
    VarChar).Value = textcountry.Text;
118.                    da.UpdateCommand.Parameters.Add("@City", SqlDbType.NVar
    Char).Value = textcity.Text;
119.                    da.UpdateCommand.Parameters.Add("@Address", SqlDbType.N
    VarChar).Value = textaddress.Text;
120.                    da.UpdateCommand.Parameters.Add("@Gender", SqlDbType.N
    VarChar).Value = genderbox.Text;
121.                    da.UpdateCommand.Parameters.Add("@Birthday", SqlDbType.N
    VarChar).Value = textbirthday.Value;
122.                    da.UpdateCommand.Parameters.Add("@NIF", SqlDbType.NVar
    Char).Value = textnif.Text;
123.                    da.UpdateCommand.Parameters.Add("@Phone", SqlDbType.NV
    arChar).Value = textphone.Text;
124.                    da.UpdateCommand.Parameters.Add("@img", SqlDbType.Image
    ).Value = img;
125.
126.                    sqlConn.Open();
127.                    da.UpdateCommand.ExecuteNonQuery();
128.                    sqlConn.Close();
129.                    disp_data();
130.                }
131.
132.                else if (dialogResult == DialogResult.No)
133.                {
134.                    MessageBox.Show("No data updated.", "Failed Update", Message
    BoxButtons.OK, MessageBoxIcon.Information);
135.                }
136.            }
137.            bool IsValidEmail(string email)
```

```
138.             {
139.                 try
140.                 {
141.                     var addr = new System.Net.Mail.MailAddress(email);
142.                     return addr.Address == email;
143.                 }
144.                 catch
145.                 {
146.                     return false;
147.                 }
148.             }
149.         }
150.         private void btndelete_Click_1(object sender, EventArgs e)
151.         {
152.             string connectionString = "Data Source=DESKTOP-
    N0B75C5;Initial Catalog=cruddatabase;Integrated Security=True";
153.             SqlConnection sqlConn = new SqlConnection(connectionString);
154.             sqlConn.Open();
155.             SqlCommand cmd = sqlConn.CreateCommand();
156.             cmd.CommandType = CommandType.Text;
157.             cmd.CommandText = "DELETE FROM cruddatabase WHERE NIF='"
     + this.textnif.Text + "'";
158.             cmd.ExecuteNonQuery();
159.             sqlConn.Close();
160.             disp_data();
161.         }
162.
163.         public void disp_data()
164.         {
165.             string connectionString = "Data Source=DESKTOP-
    N0B75C5;Initial Catalog=cruddatabase;Integrated Security=True";
166.             SqlConnection sqlConn = new SqlConnection(connectionString);
167.             SqlCommand cmd = sqlConn.CreateCommand();
168.             sqlConn.Open();
169.             cmd.CommandType = CommandType.Text;
170.             cmd.CommandText = "Select * from cruddatabase";
171.             cmd.ExecuteNonQuery();
172.             DataTable dt = new DataTable();
173.             SqlDataAdapter da = new SqlDataAdapter(cmd);
174.             da.Fill(dt);
175.             contactview.DataSource = dt;
176.             sqlConn.Close();
177.         }
178.
179.         private void contactview_CellClick(object sender, DataGridViewCellEv
    entArgs e)
180.         {
181.             if (e.RowIndex > 0)
182.             {
183.                 int rowIndex = e.RowIndex;
```

```
184.                DataGridViewRow selectedRow = contactview.Rows[rowIndex];
185.                textname.Text = selectedRow.Cells[0].Value.ToString();
186.                textmail.Text = selectedRow.Cells[1].Value.ToString();
187.                textnat.Text = selectedRow.Cells[2].Value.ToString();
188.                textcountry.Text = selectedRow.Cells[3].Value.ToString();
189.                textcity.Text = selectedRow.Cells[4].Value.ToString();
190.                textaddress.Text = selectedRow.Cells[5].Value.ToString();
191.                genderbox.Text = selectedRow.Cells[6].Value.ToString();
192.                textbirthday.Text = selectedRow.Cells[7].Value.ToString();
193.                textnif.Text = selectedRow.Cells[8].Value.ToString();
194.                textphone.Text = selectedRow.Cells[9].Value.ToString();
195.
196.                if (selectedRow.Cells[10].Value != System.DBNull.Value)
197.                {
198.                    byte[] img = (byte[])(selectedRow.Cells[10].Value);
199.                    MemoryStream ms = new MemoryStream(img);
200.                    pictureBox2.Image = System.Drawing.Image.FromStream(ms);
201.                }
202.
203.            }
204.            else
205.            {
206.                pictureBox2.ImageLocation = "C:/Users/ricar/source/repos/PGForm/
    PGForm/Resources/Generic-person.svg.png";
207.                pictureBox2.SizeMode = PictureBoxSizeMode.CenterImage;
208.            }
209.
210.            textname.Enabled = true;
211.            textmail.Enabled = true;
212.            textnat.Enabled = true;
213.            textcountry.Enabled = true;
214.            textcity.Enabled = true;
215.            textaddress.Enabled = true;
216.            genderbox.Enabled = true;
217.            textbirthday.Enabled = true;
218.            textphone.Enabled = true;
219.            textbirthday.Visible = true;
220.            btnupdate.Visible = true;
221.            btndelete.Visible = true;
222.        }
223.
224.        private void bunifuMaterialTextbox1_OnValueChanged(object sender,
    EventArgs e)
225.        {
226.            string keyword = procurarnome.Text;
227.            string connectionString = "Data Source=DESKTOP-
    N0B75C5;Initial Catalog=cruddatabase;Integrated Security=True";
228.            SqlConnection sqlConn = new SqlConnection(connectionString);
229.            sqlConn.Open();
```

```
230.                sda = new SqlDataAdapter("SELECT * FROM cruddatabase WHERE
    Name LIKE '%" + keyword + "%'", sqlConn);
231.                DataTable dt = new DataTable();
232.                sda.Fill(dt);
233.                contactview.DataSource = dt;
234.            }
235.
236.            private void btnsubmit1_Click_1(object sender, EventArgs e)
237.            {
238.
239.                if (string.IsNullOrWhiteSpace(nametxt.Text) || string.IsNullOrWhiteS
    pace(mailtxt.Text) ||
240.                    string.IsNullOrWhiteSpace(nationalitytxt.Text) || string.IsNullOrW
    hiteSpace(countrytxt.Text) || string.IsNullOrWhiteSpace(niftxt.Text))
241.                {
242.                    MessageBox.Show("You need to fill all the required fields.", "Failed
     to save new entry",
243.                        MessageBoxButtons.OK, MessageBoxIcon.Information);
244.
245.                }
246.                else if ((DateTime.Now.Subtract(birthdaypicker.Value).Days) / (365) <
     18)
247.                {
248.                    MessageBox.Show("Less than 18 Years are not allowed.", "Failed to
     save new entry",
249.                        MessageBoxButtons.OK, MessageBoxIcon.Information);
250.                }
251.
252.                else if (Validator.EmailIsValid(mailtxt.Text) == false)
253.                {
254.                    MessageBox.Show("Your e-
    mail address is invalid. Please try again.", "Failed to save new entry",
255.                        MessageBoxButtons.OK, MessageBoxIcon.Information);
256.                }
257.
258.                else if (IsValidContrib(niftxt.Text) == false)
259.                {
260.                    MessageBox.Show("Your NIF is invalid. Please try again.", "Failed
    to save new entry",
261.                        MessageBoxButtons.OK, MessageBoxIcon.Information);
262.                }
263.
264.                else
265.                {
266.                    byte[] img = null;
267.                    FileStream fs = new FileStream(imgLoc, FileMode.Open, FileAcces
    s.Read);
268.                    BinaryReader br = new BinaryReader(fs);
269.                    img = br.ReadBytes((int)fs.Length);
```

```
270.                    string connectionString = "Data Source=DESKTOP-
    N0B75C5;Initial Catalog=cruddatabase;Integrated Security=True";
271.                    SqlConnection sqlConn = new SqlConnection(connectionString);
272.                    sqlConn.Open();
273.
274.                    SqlCommand cmd = new SqlCommand("INSERT INTO cruddataba
    se (Name, Mail, Nationality, Country, City, Address, Gender, Birthday, NIF, Phone, I
    mage) VALUES ('" + nametxt.Text + "' , '" + mailtxt.Text + "' , '" + nationalitytxt.Tex
    t + "' , '" + countrytxt.Text + "' , '" + citytxt.Text + "' , '" + addresstxt.Text + "' , '" + ge
    nerbox.Text + "' , '" + birthdaypicker.Value.ToString("dd/MM/yyyy") + "' , '" + niftxt.
    Text + "' , '" + phonetxt.Text + "', @img)", sqlConn);
275.                    cmd.Parameters.Add(new SqlParameter("@img", img));
276.                    cmd.ExecuteNonQuery();
277.
278.                    sqlConn.Close();
279.
280.                    nametxt.Text = "";
281.                    mailtxt.Text = "";
282.                    nationalitytxt.Text = "";
283.                    countrytxt.Text = "";
284.                    citytxt.Text = "";
285.                    addresstxt.Text = "";
286.                    generbox.Text = "";
287.                    niftxt.Text = "";
288.                    phonetxt.Text = "";
289.                    birthdaypicker.Value = DateTimePicker.MinimumDateTime;
290.
291.                    picEmp.ImageLocation = "";
292.
293.                    MessageBox.Show("Successfully saved.", "New Entry", MessageBo
    xButtons.OK, MessageBoxIcon.Information);
294.
295.                    disp_data();
296.
297.                }
298.            }
299.
300.            private void btnprint_Click(object sender, EventArgs e)
301.            {
302.                if (printPreviewDialog1.ShowDialog() == DialogResult.OK)
303.                {
304.                    printDocument1.Print();
305.                }
306.            }
307.
308.            private void printDocument1_PrintPage(object sender, System.Drawing.
    Printing.PrintPageEventArgs e)
309.            {
310.                e.Graphics.DrawString(textname.Text, new System.Drawing.Font("Ti
    me New Roman", 14, FontStyle.Bold), Brushes.Black, new Point(100, 120));
```

```
311.                 e.Graphics.DrawString(textnif.Text, new System.Drawing.Font("Time
    New Roman", 14, FontStyle.Bold), Brushes.Black, new Point(100, 150));
312.                 e.Graphics.DrawString(textmail.Text, new System.Drawing.Font("Tim
    e New Roman", 14, FontStyle.Bold), Brushes.Black, new Point(100, 180));
313.                 e.Graphics.DrawString(textnat.Text, new System.Drawing.Font("Time
    New Roman", 14, FontStyle.Bold), Brushes.Black, new Point(100, 210));
314.                 e.Graphics.DrawString(textcountry.Text, new System.Drawing.Font("
    Time New Roman", 14, FontStyle.Bold), Brushes.Black, new Point(100, 240));
315.                 e.Graphics.DrawString(textcity.Text, new System.Drawing.Font("Tim
    e New Roman", 14, FontStyle.Bold), Brushes.Black, new Point(100, 270));
316.                 e.Graphics.DrawString(textaddress.Text, new System.Drawing.Font("
    Time New Roman", 14, FontStyle.Bold), Brushes.Black, new Point(100, 300));
317.                 e.Graphics.DrawString(genderbox.Text, new System.Drawing.Font("T
    ime New Roman", 14, FontStyle.Bold), Brushes.Black, new Point(100, 330));
318.                 e.Graphics.DrawString(textbirthday.Text, new System.Drawing.Font("
    Time New Roman", 14, FontStyle.Bold), Brushes.Black, new Point(100, 360));
319.                 e.Graphics.DrawString(textphone.Text, new System.Drawing.Font("Ti
    me New Roman", 14, FontStyle.Bold), Brushes.Black, new Point(100, 390));
320.                 e.Graphics.DrawImage(pictureBox2.Image, new Point(400, 120));
321.             }
322.
323.             private void birthdaypicker_onValueChanged(object sender, EventArgs
    e)
324.             {
325.                 if (birthdaypicker.Value == DateTimePicker.MinimumDateTime)
326.                 {
327.                     birthdaypicker.Value = DateTime.Now;
328.                     birthdaypicker.Format = DateTimePickerFormat.Custom;
329.                 }
330.                 else
331.                 {
332.                     birthdaypicker.Format = DateTimePickerFormat.Short;
333.                 }
334.             }
335.
336.             private void btnpdf_Click(object sender, EventArgs e)
337.             {
338.
339.                 using (SaveFileDialog saveFile = new SaveFileDialog()
340.
341.                 {
342.                     Filter = "PDF file|*.pdf",
343.                     ValidateNames = true
344.                 })
345.                     if (saveFile.ShowDialog() == DialogResult.OK)
346.                     {
347.                         iTextSharp.text.Document doc = new iTextSharp.text.Document(
    PageSize.A5.Rotate());
348.                         try
349.                         {
```

```
350.                     PdfWriter.GetInstance(doc, new FileStream(saveFile.FileName
    , FileMode.Create));
351.                     doc.Open();
352.                     doc.Add(new iTextSharp.text.Paragraph(textname.Text));
353.                     doc.Add(new iTextSharp.text.Paragraph(textnif.Text));
354.                     doc.Add(new iTextSharp.text.Paragraph(textmail.Text));
355.                     doc.Add(new iTextSharp.text.Paragraph(textnat.Text));
356.                     doc.Add(new iTextSharp.text.Paragraph(textcountry.Text));
357.                     doc.Add(new iTextSharp.text.Paragraph(textcity.Text));
358.                     doc.Add(new iTextSharp.text.Paragraph(textaddress.Text));
359.                     doc.Add(new iTextSharp.text.Paragraph(genderbox.Text));
360.                     doc.Add(new iTextSharp.text.Paragraph(textphone.Text));
361.                     doc.Add(new iTextSharp.text.Paragraph(textbirthday.Text));
362.                     iTextSharp.text.Image image = iTextSharp.text.Image.GetInsta
    nce(pictureBox2.Image, System.Drawing.Imaging.ImageFormat.Jpeg);
363.                     image.ScaleAbsolute(159f, 159f);
364.                     image.SetAbsolutePosition(255, 225);
365.                     doc.Add(image);
366.                 }
367.                 catch (Exception ex)
368.                 {
369.                     MessageBox.Show(ex.Message, "Message", MessageBoxButto
    ns.OK, MessageBoxIcon.Error);
370.                 }
371.                 finally
372.                 {
373.                     doc.Close();
374.                 }
375.             }
376.         }
377.
378.         public static class Validator
379.         {
380.
381.             static Regex ValidEmailRegex = CreateValidEmailRegex();
382.
383.             private static Regex CreateValidEmailRegex()
384.             {
385.                 string validEmailPattern = @"^(?!\.)(""([^""\r\\]|\\[""\r\\])*""|"
386.                     + @"([-a-z0-9!#$%&'*+/=?^_`{|}~]|(?<!\.)\.)*)(?<!\.)"
387.                     + @"@[a-z0-9][\w\.-]*[a-z0-9]\.[a-z][a-z\.]*[a-z]$";
388.
389.                 return new Regex(validEmailPattern, RegexOptions.IgnoreCase);
390.             }
391.
392.             internal static bool EmailIsValid(string emailAddress)
393.             {
394.                 bool isValid = ValidEmailRegex.IsMatch(emailAddress);
395.
396.                 return isValid;
```

```
397.             }
398.         }
399.
400.         public bool IsValidContrib(string nif)
401.         {
402.             if (!Regex.IsMatch(nif, @"\d{9}"))
403.             {
404.                 return false;
405.             }
406.
407.             char firstChar = nif[0];
408.             if (firstChar.Equals('1')
409.                 || firstChar.Equals('2')
410.                 || firstChar.Equals('5')
411.                 || firstChar.Equals('6')
412.                 || firstChar.Equals('8')
413.                 || firstChar.Equals('9'))
414.             {
415.                 int checkDigit = (Convert.ToInt32(firstChar.ToString()) * 9);
416.                 for (int i = 2; i <= 8; ++i)
417.                 {
418.                     checkDigit += Convert.ToInt32(nif[i - 1].ToString()) * (10 - i);
419.                 }
420.
421.                 checkDigit = 11 - (checkDigit % 11);
422.                 if (checkDigit >= 10)
423.                 {
424.                     checkDigit = 0;
425.                 }
426.
427.                 if (checkDigit.ToString() == nif[8].ToString())
428.                 {
429.                     return true;
430.                 }
431.             }
432.
433.             return false;
434.         }
435.     }
436. }
```