ISTEC | INSTITUTO SUPERIOR DE TECNOLOGIAS AVANÇADAS

Projeto Global

Redes Informáticas Estruturadas

Virtualização

Daniel Filipe Mendes Gonçalves

Nº8135

Orientador: Profº. Pedro Brandão

Licenciatura Informática

2017/2018

Índice

Índice de Figuras

Introdução

Primeiramente o relatório descreve de uma forma sucinta e clara as principais tecnologias utilizadas no projeto. O mesmo recai sobre a virtualização e *Cloud Computing* e as suas principais funções, numa primeira fase é feita uma abordagem teórica sobre as vantagens e desvantagens quando aplicadas em casos práticos. A virtualização no decorrer dos dias de hoje tem sido uma mais valia para as empresas, sendo versátil e económica, permitindo uma redução de equipamentos *on premises,* tornando também uma fácil gestão da rede.

Para além do estado da arte encontra-se também descrito todo o processo de desenvolvimento do projeto, tendo por base os pontos mais significantes para a sua elaboração. O projeto conta com a construção de uma rede estruturada, suportada por um *host*, o mesmo irá conter três servidores sendo um deles o *domain controler*, a infraestrutura terá também um cliente conectado há rede e acesso a aplicação desenvolvida em *Ado.net*.

Resumo

O intuito deste projeto é criar uma solução de baixo custo para pequenas e médias empresas na área do retalho no âmbito da virtualização e *Cloud Computing*. Será demonstrada numa primeira fase uma visão geral das tecnologias usadas no contexto do projeto apresentado, é feita uma abordagem teórica sobre os pontos fortes e as desvantagens que poderá ocorrer da sua aplicação num uso prático. Após abordagem teórica é apresentado o desenvolvimento do projeto, que consiste na construção de uma rede estruturada suportada por três servidores num *host*, existindo um controlador de domínio com as ferramentas de ADDS, DHCP, NPS, DNS, tendo também um servidor de Routing server que permite apenas uma ligação ao exterior, e um SQL Server que irá conter os dados das compras e dos clientes. O cliente estará também configurado no *host*, terá permissões da gestão da base de dados e acesso ao ponto de venda. Na descrição dos processos efetuados para a configuração dos servidores estarão presentes os passos mais relevantes em cada nó da estrutura que constitui a rede.

A aplicação *Point of Sale* – Barrigas do Asfalto, é baseada num ponto de vendas em que os clientes irão selecionar os itens que pretendem adquirir, adicionando a um carrinho de compras. Estes terão também, permissões sobre os dados lançados na base de dados, podem efetuar o pagamento através de um dos separadores da aplicação que gera o recibo com toda a informação do cliente em formato PDF. A aplicação é desenvolvida em *C#* e utiliza a tecnologia ADO.net com ligações a SQL *Server*.

Palavras Chave: *Hyper-V;* Virtualização*; Datacenter; Cloud Computing;* Redes Empresariais*; Windows Server.*

Abstract

The aim of the project is to create a low cost solution for small and medium businesses of the selling area through virtualisation and cloud computing. In the first stage it will be shown a global view in the technologies used on the project, there is a theoretical approach on the strength and weakness that may happen when applied in a practical way. After the theoretical approach it is shown the project's development, mainly consisting in the construction of a structural network supported by three servers in a host, existing a controller of the domains ADDS, DHCP, NPS, DNS, and also a routing server that only allows a single connection to the exterior and a SQL server that will hold the selling and clientele data. The client will also be in the host and will have permission to manage the database and access the selling. In the description of the process used in the configuration of the servers it will be present the main steps in each knot of the network structure.

The application Point of Sale – " Barrigas do Asfalto", is based in a selling system where the clients will select the items they pretend to purchase, adding them to the selling cart. It will also have permission over the data in the database, where they can make the payment through another part of the application that will emit the invoice via PDF format. The application is developed in C# and uses ADO.net technology with SQL Server connection.

Keywords: Hyper-V; Virtualization; Datacenter; Cloud Computing; Enterprise Networks; Windows Server.

Estado da Arte

Virtualização

A virtualização surgiu na década de 60 e teve como conceito o *Time Sharing*, onde apenas existiam *Mainframes*[1]. O uso de computadores pessoais era escasso, nesta altura tornava-se muito complicado efetuar a instalação de um novo programa ou atualizar, visto que implicava uma alteração completa do *hardware*. A evolução da virtualização permitiu utilizar com o mesmo *hardware* outros sistemas e repartir melhor os recursos entre as aplicações. (Microsoft, 2015, p.8)

Na década de 80, à medida que os computadores começaram a tornar-se mais comuns visto terem uniformizado o seu *hardware*, a quantidade de sistemas operativos convergiu em três famílias (Unix, Macintosh, Microsoft). Neste contexto a utilização de máquinas virtuais perdeu alguma importância devido às pessoas terem pela primeira vez a possibilidade de ter um computador pessoal como também de optar por um sistema operativo. Mais tarde, com melhoramento dos processadores, a disseminação dos sistemas distribuídos e a omnipresença das redes de computadores causaram várias razões para o ressurgimento da virtualização. (Carissimi, 2009)

Com o avançar da tecnologia e da virtualização é possível economizar bastante dinheiro quanto a gastos de energia, espaço físico e melhor aproveitamento dos recursos dos equipamentos. A virtualização é um método para abstrair recursos físicos da maneira como eles interagem com outros recursos. Por exemplo, se abstrair o *hardware* físico do sistema operacional, é possível obter o benefício de poder mover o sistema operacional entre diferentes sistemas físicos. Existem outras formas de virtualização disponíveis como a apresentação virtualização, virtualização de *desktops* e virtualização de aplicativos. (Penek, 2015, p.9)

A virtualização de servidores permite que vários servidores sejam executados no mesmo servidor físico. O *Hyper-V* é uma ferramenta de virtualização de servidor que permite que se mova uma máquina física para um ambiente virtual, permitindo e gerindo-

[1] Mainframe- É um computador de grande porte dedicado normalmente ao processamento de um volume enorme de informações.

a num único servidor. Com este método é possível consolidar servidores físicos. Na virtualização de apresentação as aplicações são executadas num computador diferente e somente as informações da tela são transferidas para o computador. Um exemplo de virtualização de apresentação é o *Microsoft Remote Desktop Services* no *Windows Server 2012 R2*. Na virtualização da área de trabalho é fornecida uma máquina virtual no computador, comparável à virtualização do servidor. Neste caso as aplicações são previamente instaladas no servidor e o cliente apenas necessita de ter o sistema operativo instalado. Um exemplo desta forma de virtualização é o *Microsoft Virtual PC*. A virtualização de aplicativos previne conflitos entre as aplicações instaladas localmente no computador e as que se encontram no servidor. Também ajuda a reduzir as incompatibilidades e as necessidades de teste de aplicação para aplicação. Um exemplo de uma ferramenta de virtualização de aplicativos é o *Microsoft Application Appliance Virtualization* (App-V). (*ibidem*. pp.440-441)

Windows Server 2016

Segundo Stanek (2016), *o Windows Server 2016* é a última versão lançada pela Microsoft para os servidores. Esta versão inclui opções melhoradas de *deployment* e *management* que não estavam nas versões anteriores. Algumas dessas melhorias foram: *Windows Server Conteires, Hyper V Conteires, Nano Server, Storage Spaces Direct* e *Storage Replica.* O autor refere que *Windows Server* se assemelha bastante ao *Windows 10*, tanto a nível gráfico como de definições de segurança, rede e *storage*. Na Figura 1 e 2 são apresentadas algumas das semelhanças entre estes dois sistemas (*Windows Server 2016* e *Windows 10*).

Figura 1: Novo Painel de Controlo Windows Server 2016

Fonte: (Microsoft, 2016).

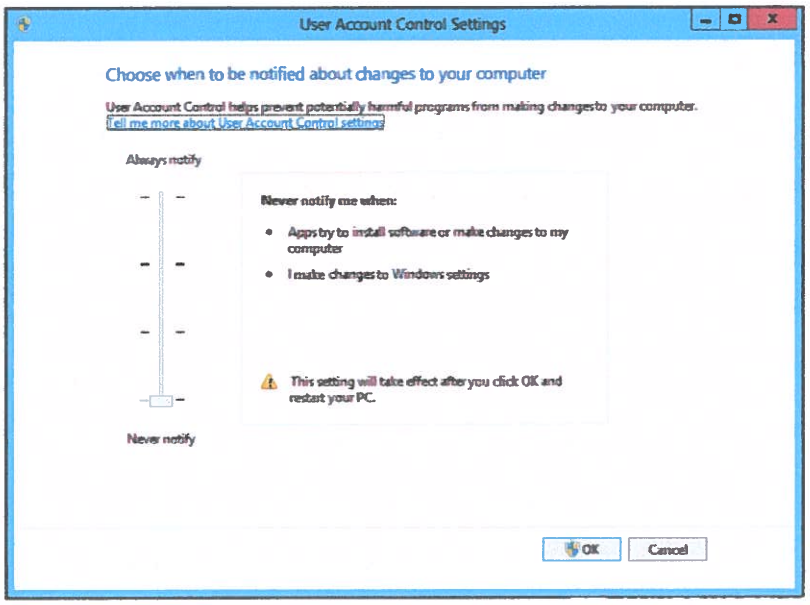


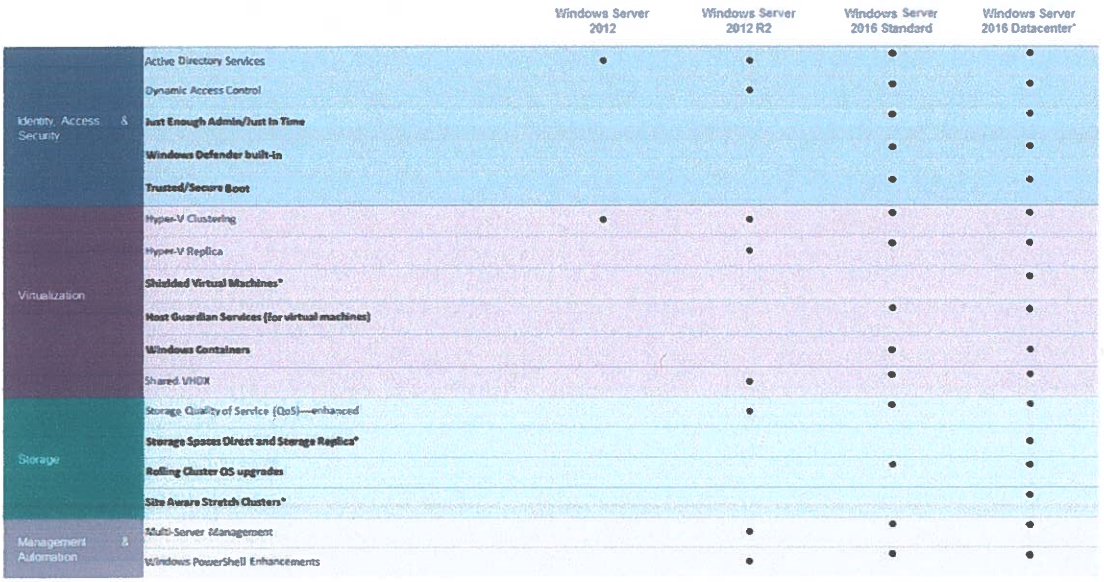
Figura 2: UAC - User Account Control Settings

Fonte: (Microsoft, 2016).

O *Windows Server 2016* está disponível em duas versões (*Standard* e *Datacenter*), ambas as versões têm os mesmos requisitos a nível de *hardware*. Os requisitos são: um processador a 64bits[2] que suporte NX e DEP, no mínimo 512 MB RAM para a instalação

[2] 64bits – Tipo de arquitetura do processador.

mínima e, pelo menos 2GB para uma versão *Standard,* necessário também 32GB de espaço Livre e uma porta *Ethernet Gigabit*[3]. (Stenek, 2016, pp. 1-4*)*



		Windows Server 2012	Windows Server 2012 R2	Windows Server 2016 Standard	Windows Server 2016 Datacenter*
Identity, Access & Security	Active Directory Services	•	•	•	•
	Dynamic Access Control		•	•	•
	Just Enough Admin/Just In Time			•	•
	Windows Defender built-in			•	•
	Trusted/Secure Boot			•	•
Virtualization	Hyper-V Clustering	•	•	•	•
	Hyper-V Replica		•	•	•
	Shielded Virtual Machines*				•
	Host Guardian Services (for virtual machines)			•	•
	Windows Containers			•	•
	Shared VHDX		•	•	•
Storage	Storage Quality of Service (QoS)—enhanced		•	•	•
	Storage Spaces Direct and Storage Replica*				•
	Rolling Cluster OS upgrades			•	•
	Site Aware Stretch Clusters*				•
Management & Automation	Multi-Server Management		•	•	•
	Windows PowerShell Enhancements		•	•	•

Figura 3: Diferença entre versões do Windows Server

Fonte: (Infordisa, 2016).

Hyper-V

O *Hyper-V* é uma *role* do *Windows Server* que surgiu a partir do *Windows Server 2008* que fornece e permite aos administradores criar e gerir ambientes virtuais aproveitando a tecnologia do *Windows Server*. Este *Hypervisor* da *Microsoft* permite correr várias instâncias em simultâneo no mesmo servidor, todos os recursos alocados às máquinas virtuais são estipulados previamente antes da configuração do sistema operativo. Algumas das funcionalidades disponíveis no *Hyper*-v são os sistemas operativos com arquitetura de *32bits* e *64bits* que podem correr em simultâneo no *HyperV*. Contudo, podem também ser adicionados outros sistemas operativos para além do Windows. A Migração Rápida ou *Fast Migration* permite correr várias máquinas virtuais num ambiente de *cluster*, de forma a reduzir o tempo de inatividade obtendo assim um serviço de alta disponibilidade. Outra funcionalidade é a memória dinâmica, uma forma que auxilia a gestão de memória física mais eficientemente, permitindo

[3] Gigabit -é o termo que descreve várias tecnologias para transmissão de quadros em uma rede a uma velocidade de 1 Gigabit por segundo.

também uma melhor distribuição dos recursos pelas máquinas virtuais. O isolamento de rede é a capacidade do *Hyper-V* manter as redes virtuais isoladas da infraestrutura da rede física do servidor principal. (Microsoft, 2015 p.9):

Processor/Memory Feature	Windows Server 2008 R2	Windows Server 2012
Logical processors on hardware	64	320
Physical memory	1 TB	4 TB
Virtual processors per host	512	2,048
Virtual processors per virtual machine	4	64
Memory per virtual machine	64 GB	1 TB
Active virtual machines	384	1,024
Maximum cluster nodes	16	64
Maximum cluster virtual machines	1,000	8,000

Figura 4: Comparação entre Hyper-V 2008 e 2012

Fonte: (Arnavsharma, 2014).

Não é possível considerar o *Hyper-V* como a solução para todos os problemas de TI, porém dependendo das circunstâncias poderão existir desvantagens. A subcarga é considerada uma desvantagem no caso do próprio computador (servidor) não existindo limitação na quantidade de máquinas virtuais e existindo um equilíbrio entre ambas. Se existir uma vulnerabilidade de segurança no VMM, todas as máquinas virtuais poderão ter problemas. A portabilidade das máquinas pode ser uma desvantagem caso seja necessário exportar de AMD-V para um equipamento Intel. Ter um plano de contingência é sempre uma boa política, ou seja, caso a máquina principal falhe ter como *backup* uma máquina pronta como servidor, pois todas as outras estarão comprometidas. No que diz respeito à virtualização nem todas as aplicações podem ter um bom desempenho, porém é necessário avaliar a solução antes de ser implementada, já que poderão também existir gastos não previstos com a mão-de-obra, manutenção, formação ou implementação. (Desai, 2012)

Virtual Network Switch

O *Virtual Network Switch* é uma ferramenta que permite criar ligações entre máquinas virtuais de acordo com o pretendido. As redes virtuais no *Hyper-V* são fornecidas de forma segura e dinâmica porque, podem ser definidos parâmetros de redes virtuais em grande escala como em pequena escala. Este tipo de funções poderá ser útil na separação de redes e no que realmente se pretende para a gestão da rede (Stenek, 2015 p.454). Toda a gestão da rede virtual passa pelo *Virtual Switch Manager* como se verifica através da próxima Figura.

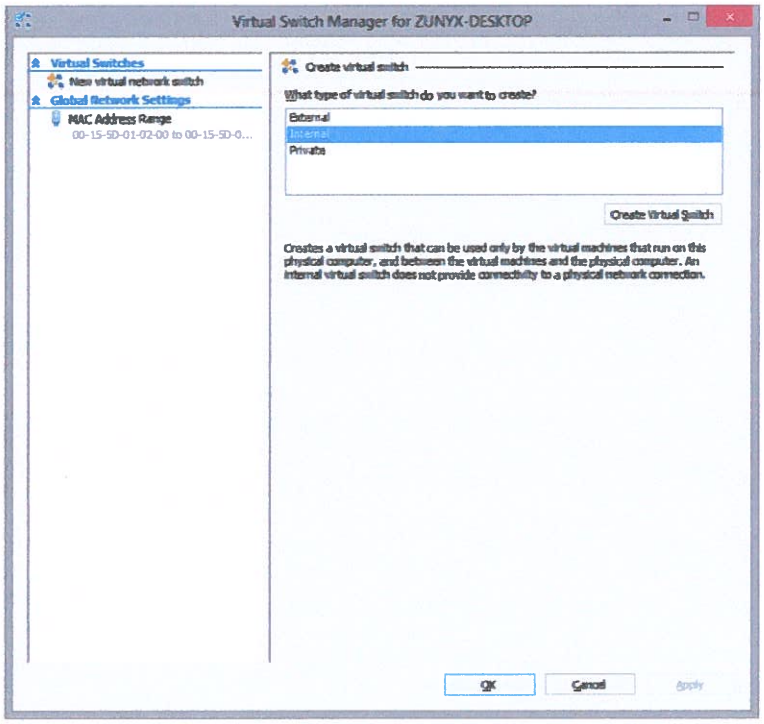


Figura 5: Virtual Network Manager

Fonte: (Microsoft,2017).

Ao utilizar o *Virtual Network Manager* é permitido o acesso a inúmeras funções desde criar, configurar e eliminar redes virtuais. Os principais tipos de redes possíveis de criar são: a ligação externa, interna e a privada. A ligação externa permite que qualquer máquina virtual ligada a este tipo de rede tenha acesso a rede física, por norma esta opção é utilizada quando se pretende que um dos servidores seja acedido por terceiros. A configuração da placa virtual em modo *External* é bastante utilizada para ambientes de produção. A ligação *Internal* permite que as máquinas comuniquem entre si e até mesmo com *host,* porém não existe comunicação com a rede física. Com a placa em modo *Private*

é possível comunicar entre máquinas virtuais, contudo fica barrada a comunicação com o *Host* e com a rede física (*ibidem*, 2015 p.454) .

Active Directory

O *Active Directory* tem como objetivo centralizar os utilizadores, computadores e todos os comportamentos da infraestrutura *Windows*. As informações armazenadas determinam quais os recursos acessíveis aos utilizadores, através do uso de permissões atribuídas aos objetos do *Active Directory.* Assim, é possível controlar todos os aspetos da segurança da rede (Microsoft, 2016).

Penek (2015, p. 7) refere a ferramenta *Active Directory* como uma peça fundamental no domínio Microsoft: *"Active Directory is the heart and soul of a Microsoft domain, and I can never talk enough about the roles and features included with Active Directory"*.

Os serviços de Certificado do *Active Directory (*AD/CS*)* fornecem um conjunto customizável de serviços que permitem gerir a infraestrutura de chave pública de certificados (PKI)[4]. Estes Certificados podem ser usados em segurança de *software* que empregam tecnologias de chave pública Serviços de Domínio. Os serviços de domínio *Active Directory (*AD/DS*)* incluem novos recursos que tornam a implementação de controladores de domínio mais simples e permite implementá-los de forma mais rápida. O AD/DS também torna os controladores de domínio mais flexíveis, tanto para auditar como para autorizar o acesso a arquivos. Além disso, o AD/DS foi projetado para facilitar a execução de tarefas administrativas através de experiências de gestão de gráficos e *scripts* consistentes. Nos serviços de implementação do Windows o administrador pode instalar de forma remota aplicações e sistemas operativos, isto é, poderá configurar novos computadores usando uma instalação a partir de uma imagem na rede. (*ibidem*, 2015 p.700).

---

[1] *Host* - Servidor ou Computador que serve de alojamento, neste caso servidor onde estão alojadas as máquinas virtuais.

[4] *Public Key Infrastructure* – Tem como objetivo verificar a identidade de um utilizador/entidade com mais confiança.

ADO.NET

O *ADO.NET* (*ActiveX Data Objects*) é a principal biblioteca de acesso a dados da Microsoft para programadores .NET e é considerada o core de muitas tecnologias focadas em gestão de dados. Esta *framework* trabalha juntamente com *C#* é permite aos programadores efetuarem ligações com bases de dados (*SQL Server*) de uma forma mais simples e rápida. (Patrick, 2010, p XVII)

Através da sua utilização, permite ao programador executar comandos a partir do código sendo processados ou colocados diretamente num objeto *ADO.NET (Dataset),* permitindo assim efetuar um conjunto de tarefas podendo assim manipular os dados dentro da base de dados. Algumas das principais características do *ADO.NET* são: um sucessor do ADO mais flexível, é um sistema desenhado para ambientes desconectados, ou seja, que não esteja sempre em contacto com a base dados, contém suporte avançado para XML, contém um conjunto de classes, interfaces, estruturas e enumerações que fazem a gestão do acesso a dados dentro da *framework.* A evolução do ADO para *ADO.net* foi radical no sentido em que começou por utilizar apenas 3 objetos (*Connection*, *Command* e *Recordset*), passando a utilizar (*Connection*, *Transaction*, *Command*, *DataSet*, *DataReader* e *DataAdapter*). (Ferreira, 2004, pp.5-11)

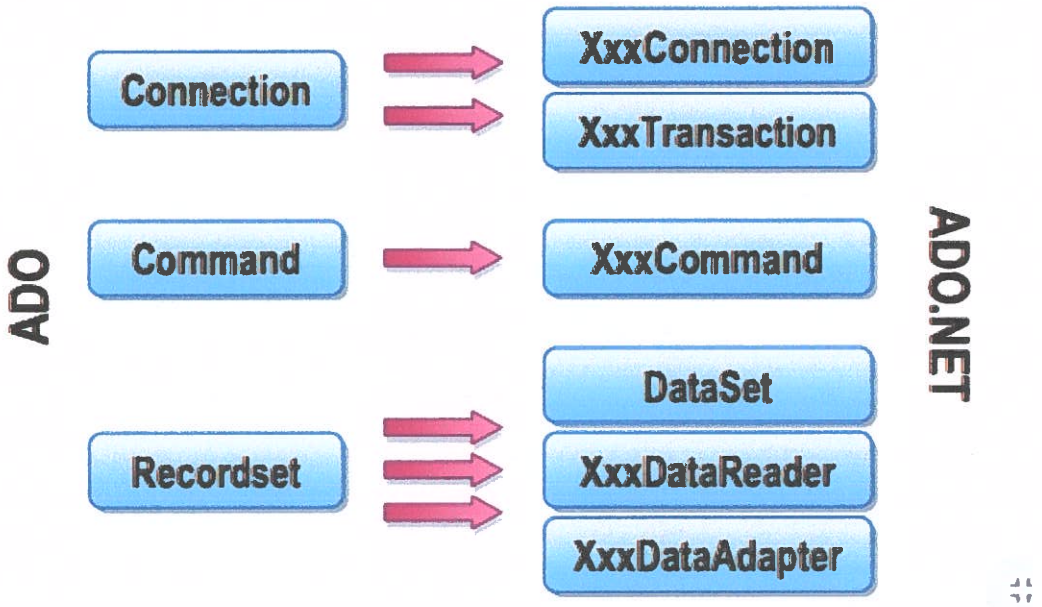


Figura 6: Evolução do ADO para ADO.NET

Fonte: (Ferreira, 2004, p 11).

SQL

O SQL, também conhecido como *Strutured Query Language,* trata-se de uma linguagem padrão para o tratamento de dados, gestão de bases dados e o seu relacionamento. Como a evolução dos sistemas operativos a linguagem SQL teve também a sua evolução desde muito cedo. Em 1974 dá-se a criação da linguagem em laboratório pela IBM, em 1989 a Microsoft cedeu à Sybase permissão para utilizar o seu código e criar uma ferramenta para a gestão de base de dados, esta versão conhecida como SQL *Server 1.0* (Filipi). Ao longo dos anos e, até à atualidade, foram desenvolvidas diversas versões do SQL, a própria linguagem mantém essencialmente declarativa. Alguns dos principais comandos SQL para manipulação de dados são: *INSERT* (inserção), *SELECT* (consulta), *UPDATE* (atualização), *DELETE* (exclusão). *SQL* possibilita ainda a criação de relações entre tabelas e o controle do acesso aos dados (Calbimonte, 2018).

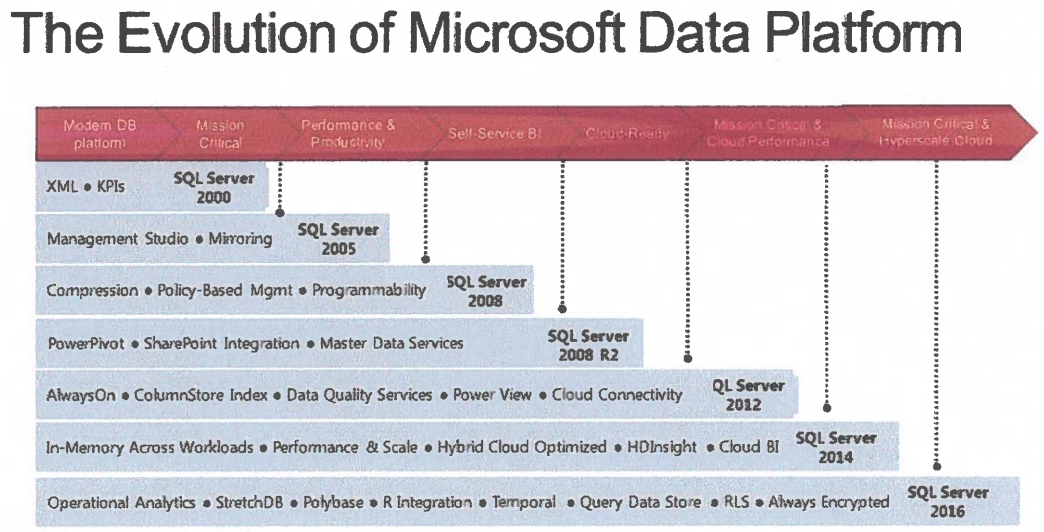


Figura 7: Evolução dos SQL Server

Fonte: (Sqlmvp, 2016).

Recentemente foi lançada uma versão melhorada para a gestão de base de dados da Microsoft, o *SQL Server 2017*. Esta versão veio aprimorar ainda mais o funcionamento de versões anteriores permitindo escolhas de idiomas de desenvolvimento, tipos de dados, instalações ou *cloud* e sistemas operacionais, liberdade de instalação do *software* em ambiente *Linux* e uma melhor otimização dos recursos (Microsoft, 2017).

As versões disponíveis do SQL *Server 2017* são: a *Enterprise* que oferece capacidades de *Datacenter* com um desempenho bastante rápido, permitindo a

virtualização ilimitada e oferecendo níveis de serviços de alta disponibilidade para os utilizadores finais. A *Standard* fornece uma gestão de base de dados básica que se adapta a departamentos e pequenas organizações para executar as suas aplicações e, oferece suporte em ferramentas de desenvolvimento comuns para o local e para a *Cloud*. A *Web* é uma versão de baixo custo que fornece capacidades de escalabilidade, acessibilidade e capacidade de gestão para ambientes *Web* de pequena a grande escala. A *Developer* permite aos programadores desenvolver qualquer tipo de aplicação no topo do *SQL Server* pois inclui todas as funcionalidades da edição *Enterprise*, no entanto, é licenciada para uso como um sistema de desenvolvimento e teste. Por último, a *Express Edition* é uma versão gratuita, ideal para aprender a criar aplicações *desktop* e para pequenos servidores de dados. Esta é a melhor escolha para programadores independentes de *software*, que desenvolvem e criam aplicações para clientes (*ibidem*, 2017).

Feature	Enterprise	Standard	Web	Express with Advanced Services	Express
Maximum compute capacity used by a single instance - SQL Server Database Engine[1]	Operating system maximum	Limited to lesser of 4 sockets or 24 cores	Limited to lesser of 4 sockets or 16 cores	Limited to lesser of 1 socket or 4 cores	Limited to lesser of 1 socket or 4 cores
Maximum compute capacity used by a single instance - Analysis Services or Reporting Services	Operating system maximum	Limited to lesser of 4 sockets or 24 cores	Limited to lesser of 4 sockets or 16 cores	Limited to lesser of 1 socket or 4 cores	Limited to lesser of 1 socket or 4 cores
Maximum memory for buffer pool per instance of SQL Server Database Engine	Operating System Maximum	128 GB	64 GB	1410 MB	1410 MB
Maximum memory for Columnstore segment cache per instance of SQL Server Database Engine	Unlimited memory	32 GB[2]	16 GB[2]	352 MB[2]	352 MB[2]
Maximum memory-optimized data size per database in SQL Server Database Engine	Unlimited memory	32 GB[2]	16 GB[2]	352 MB[2]	352 MB[2]
Maximum memory utilized per instance of Analysis Services	Operating System Maximum	Tabular: 16 GB MOLAP: 64 GB	N/A	N/A	N/A
Maximum memory utilized per instance of Reporting Services	Operating System Maximum	64 GB	64 GB	4 GB	N/A
Maximum relational database size	524 PB	524 PB	524 PB	10 GB	10 GB

Figura 8: Limites Máximos do SQL Server 2017.

Fonte: (Microsoft, 2018).

Discussão

A virtualização tornou-se uma tecnologia revolucionária que proporcionou às empresas uma gestão dos recursos tendo em conta a sua área de negócio. Com recurso a *softwares* que apoiam a criação de máquinas virtuais, em ambientes centralizados, tornou-se ainda mais fácil a sua utilização. O desempenho do *hyper-v* sobressai sobre outras ferramentas do ramo como, a sua simplicidade em gerir a memória de cada máquina virtual. Com ajuda da ferramenta *Virtual Network Switch* é possível que as empresas criem sub-redes (*vlans*) e, a partir da utilização de placas internas, externas e privadas, efetuem uma melhor gestão de redes. A utilização da *framework ADO.net* da Microsoft em C# e a sua extensa biblioteca de ferramentas é considerada uma das melhores para o desenvolvimento de aplicações Windows.

Contextualização

A virtualização tem vindo a ser cada vez mais aperfeiçoada para obter melhores ganhos e eficiência, tornando-se cada vez mais fácil ter servidores na *Cloud* ou até mesmo *On Promise*. Para as pequenas e médias empresas esta tecnologia permite tirar melhor partido do hardware e ter uma redução a nível de custos. A *Cloud* está cada vez mais a tornar-se uma boa opção para assegurar os dados das empresas, existindo *datacenters* com o objetivo de vender recursos *hardware*, contudo os preços praticados não estão ao alcance de todos, por isso continuam a recorrer à utilização servidores locais. Nos dias de hoje a *Cloud* oferece muito mais que uma máquina na rede pois, todo core onde estão alojadas as máquinas usufruem de uma proteção acima do que o que se consegue garantir em *on promise*, o que garante aos utilizadores maior segurança dos seus dados com a possibilidade de recuperação dos dados quase imediata. A Figura 10 representa o diagrama do projeto que irá ser apresentado, sendo uma opção de baixo custo para as empresas e será composta por um *Host* (computador local), que irá utilizar a ferramenta Hyper-*V* para configurar três servidores e uma máquina cliente. Será configurado um controlador de domínio que irá incluir a ferramenta NPS (*Network Policie Server*), DHCP e DNS, que por sua vez terá auxilio de um *Routing Server* pois será o único ponto de comunicação para a rede exterior. Para além dos anteriores será adicionado um servidor SQL, que irá suportar a base de dados para os clientes.

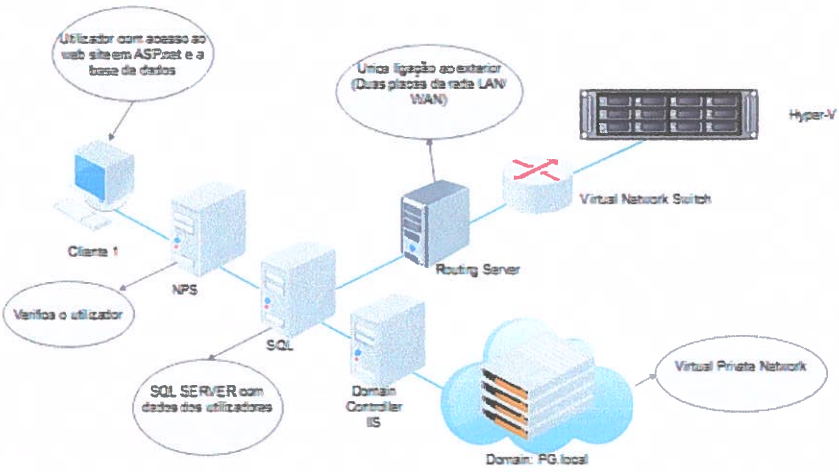


Figura 9: Diagrama do Projeto

Fonte: (Pedro Brandão, 2017).

Desenvolvimento

A elaboração deste projeto desenvolveu-se nos seguintes passos:

1. Criar partição para efetuar *dual boot* do sistema operativo (*Server Windows 2016 Standard*) na máquina física.
2. Instalação e configuração da *Hyper-V* no servidor físico.
3. Após instalação do *Hyper-V*, iniciar a instalação do *Domain Controller* e criar o domínio (PG.local).
4. Instalação e configuração do Routing Server (*Server Windows 2016 Standard*).
5. Criar máquina virtual que irá servir de *SQL Server*, onde será instalada uma instância de *SQL Server* que irá suportar a base de dados.
6. Configuração da role *NPS* no *Domain Controller*
7. Instalação do computador cliente.
8. Desenvolvimento da aplicação em ADO.net.
9. Publicação do programa no computador cliente.

Posteriormente será demonstrada de uma forma sucinta e clara como foi elaborado o projeto, contando com os pontos mais relevantes para a sua construção.

Esquema Detalhado

Para a execução do projeto foi necessário reunir previamente os dados que serão utilizados na rede, como os *ip's* que serão atribuídos aos servidores, licenças Microsoft e a própria especificação.

Contas

Na construção do projeto os acessos criados foram:

- PG.local\Administrator – Password01 //DC1 / DC2 / Routing Server
- PG.local\Administrator – Password01 //SQL Server
- SRVSQL\SQLSERVERPG – User: Sa – Sql2016 //SQL Server
- PG.local\Cliente1 – Password01 //Cliente1

Máquinas – Rede/Função/Espaço/SO/Licença

- SRVDC01 (*Domain Controller*)
  192.168.2.1 *LAN*
  Windows Server 2012 Standard
  Key: 33BXN-2FQMG-QV4QY-2VH8W-82Q9R
  Espaço: 50GB

- SRV-RSERVER (*Routing Server*)
  192.168.2.2 LAN
  192.168.1.154 WAN (*DHCP*)
  Windows Server 2016 Standard
  Key: 33BXN-2FQMG-QV4QY-2VH8W-82Q9R
  Espaço: 50GB

- SRVSQL (Servidor *SQL Server*)
  192.168.2.4 *LAN*
  Windows Server 2016 Standard
  Key: 33BXN-2FQMG-QV4QY-2VH8W-82Q9R
  Espaço: 50GB
- CL1 (Cliente 1)
  192.168.2.11 (DHCP)
  Windows 7 Pro x86
  Key: M4NCB-TDGTW-2FQMG -3RKGP-7CCDH

Instalação do *Host*

A instalação do *Host* é o passo mais importante para a elaboração deste projeto, pois possibilitará instalar a ferramenta *Hyper-V* que fornece a possibilidade de instalar múltiplos sistemas operativos no mesmo *Host*.

Para a instalação do *Windows server 2016 Standard* efetua-se uma preparação prévia do disco onde será instalado, pois será utilizado um computador com um sistema operativo já instalado.

Dentro das ferramentas administrativas existe um atalho com a designação de gestão de computadores, seleciona-se no painel lateral esquerdo o gestor de discos. Neste menu, irá ser possível visualizar todos os discos locais e terá que se selecionar o disco que se pretende reduzir e pressiona-se com o lado direito do rato (Figura 10).

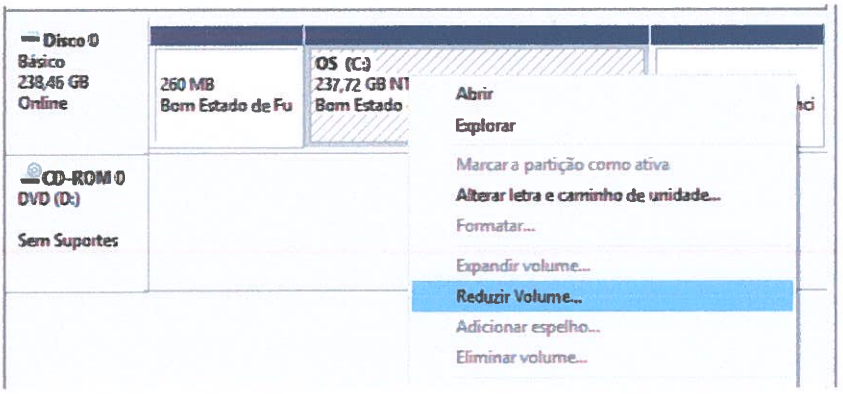


Figura 10: Gestor de discos.

Fonte: do autor.

Seguidamente, irá ser criada uma partição de 50Gbs para o *Host* (*Windows Server 2016*), quanto aos restantes requisitos serão os da máquina que está a utilizar (Processador AMD *Phenom X4 3.20Ghz*, 8GB de *Ram, 50GB* SSD, 1TB disco externo).

Após a redução do espaço, avança-se com a instalação do sistema operativo, coloca-se o CD de instalação e reinicia-se o computador. Ao pressionar a tecla F8 no arranque surge um o menu que permite escolher o dispositivo de arranque, neste caso seleciona-se arranque a partir drive de CD. No processo de instalação do sistema operativo escolhe-se o disco de 50 *Gbs* que anteriormente foi preparado para receber o *Windows Server 2016.*

Servidor *Hyper-V*

Após a instalação do *host* inicia-se a instalação da role *Hyper-V*. O *host* é o nome que se dá a uma estrutura em ambiente virtual, com esta role é possível suportar vários sistemas operativos em simultâneo, podendo assim gerir os recursos de cada umas das máquinas que serão instaladas.

Para efetuar a instalação do *Hyper-V* é necessário dentro do *Server Manager* selecionar a opção *Add roles and features* e, em seguida, no separador *Server Roles* escolhe-se a opção *Hyper-V* como se pode verificar na Figura 11.

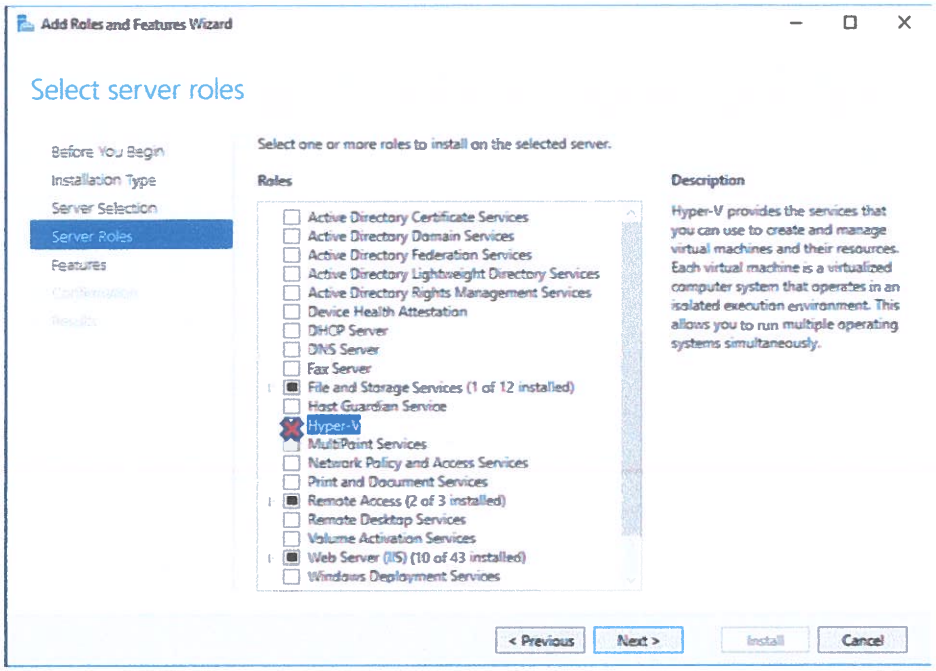


Figura 11: Hyper-V Role.

Fonte: do autor.

Após instalação da *role Hyper-V* poderá ser necessário reiniciar o servidor para que sejam aplicadas corretamente as definições, sendo necessário garantir sempre que o Windows está atualizado.

Avançando para a configuração do *Virtual Switch Manager*, esta ferramenta permitir criar e configurar várias ligações de rede, separando as ligações. Dentro do *Hyper-V* seleciona-se no canto superior direito a opção *Virtual Switch Manager* e como já existe uma placa configurada apenas será necessário garantir que essa placa tem ligação externa. Neste caso, renomeia-se a placa de rede para *External Network Project* o que facilitará identificar dentro das máquinas virtuais a rede (Figura 12). Em seguida escolhe-se a opção New *Virtual Network Switch* e adiciona-se a placa interna *Internal Network Switch*, sendo necessário escolher a opção *internal network* (Figura 13).

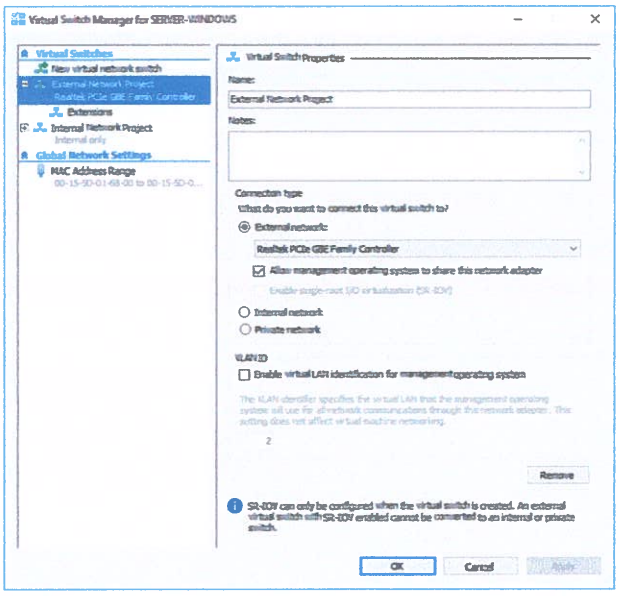


Figura 12: External Network (WAN).

Fonte: do autor.

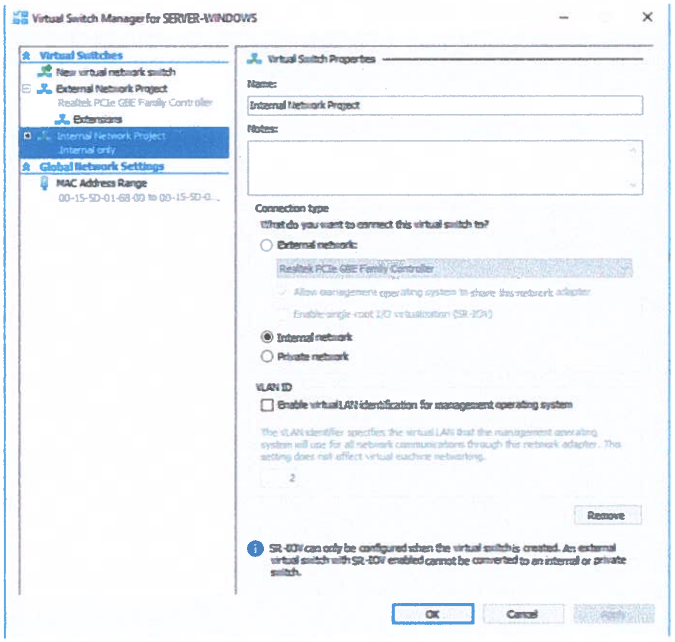


Figura 13: Internal Network (LAN).

Fonte: do autor.

Com a configuração do *Virtual Switch Manager* dá-se início à configuração dos restantes servidores.

Máquinas Virtuais

Este tópico irá abordar a instalação das máquinas virtuais, e o procedimento será o mesmo para todas (SVRDC01, SRV-RSERVER, SRVSQL, CL1). As características selecionadas foram:

- Armazenamento

Servidores: 50GB de disco;
Máquina Cliente: 40GB de Disco;

- Placas de Rede

O servidor SRV-RSERVER irá possuir 2 placas de rede (*Internal* e *External*); Os restantes servidores e clientes iram apenas possuir a placa de rede *Internal*, que irá estar interligada com o domínio PG.local;

- Sistema Operativo

Servidores: *Windows Server 2012 R2 Standard* (SRVD01);
Servidores: *Windows Server 2016 Standard*;
Máquina Cliente: *Windows 7 Professional*;

No Gestor do *Hyper-v*, clica-se em *Actions, New* e depois *Virtual Machine*, e em seguida dá-se o nome adequado à máquina que foi mencionado anteriormente (ex. SVRDC01). A instalação da máquina é feita num disco à parte devido ao pouco espaço do disco *SSD*. Seleciona-se a geração 2 e avança-se para a alocação da *RAM* (2048MB) em modo dinâmico (Figura 14).

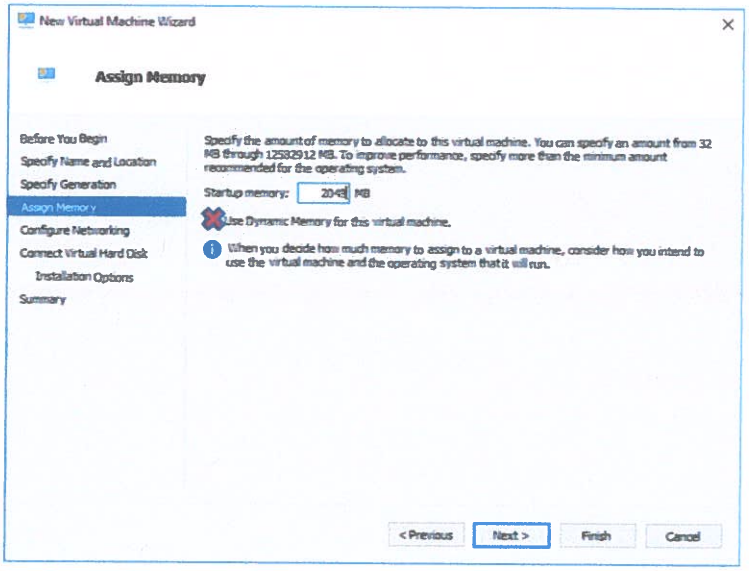


Figura 14: Atribuição de Memória.

Fonte: do autor.

Após a configuração da memória avança-se para a configuração de rede onde será necessário optar pelo adaptador adequado ao servidor a instalar (*Internal* ou *External*). Seleciona-se a *Internal* visto que o único computador com saída para a *internet* direta será o *Routing Server*, define-se em seguida a localização do *VHDX* e por último o tamanho do disco (50GB).

É necessário agora, selecionar a imagem (*.iso*) do *Windows Server 2016* ou 2012 que irá iniciar no primeiro arranque da máquina (Figura 15).

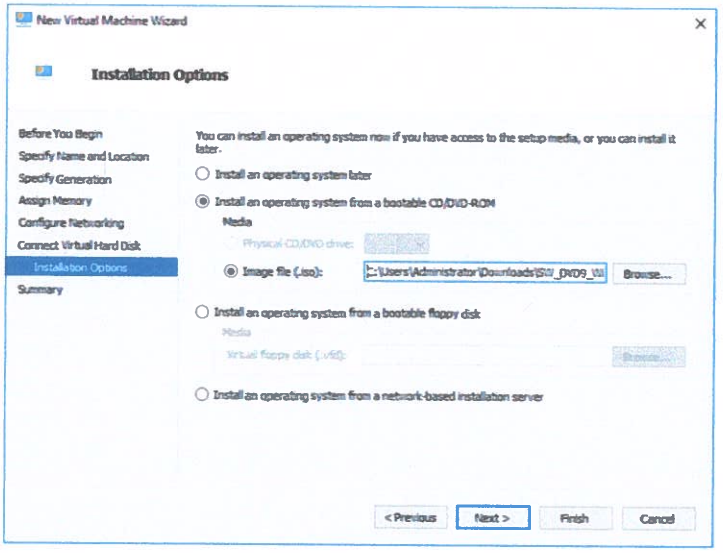


Figura 15: Operating system from a bootable CD/DVD.

Fonte: do autor.

Finalmente, após a conclusão das opções de instalação da máquina virtual dá-se início à configuração do sistema operativo. Durante a instalação é necessário escolher a versão *Standard* (*Desktop Experience*) para ter acesso à parte gráfica do Windows (Figura 16). Posteriormente, escolhe-se a partição criada (50GB) e termina-se.

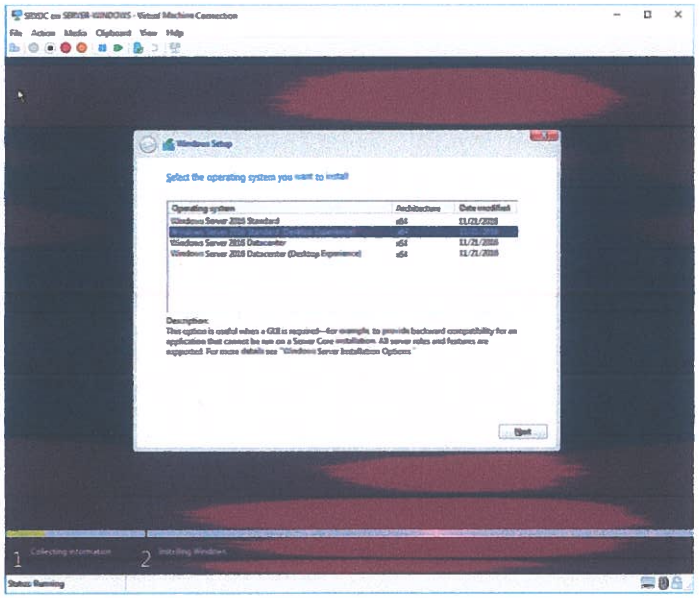


Figura 16: Versão do Sistema Operativo.

Fonte: do autor.

Repete-se este procedimento para as restantes máquinas.

Configuração do *Domain Controller* (SRVDC01)

Para a configuração do servidor de domínio é necessário seguir os seguintes passos:

1º Nome do Servidor: SRVDC01

2º Configuração *IPv4*:

IP: 192.168.2.1

Máscara: 255.255.255.0

Gateway: 192.168.2.254

DNS: 127.0.0.1

3º Desativar *IPv6*.

4º Ativar os Serviços de *ICMP* na *Firewall*.

Dentro do *Server Manager* seleciona-se a opção *Add roles and features* e no separador *Server Roles* seguido da opção *Active Directory Domain Services (Figura 17)*.

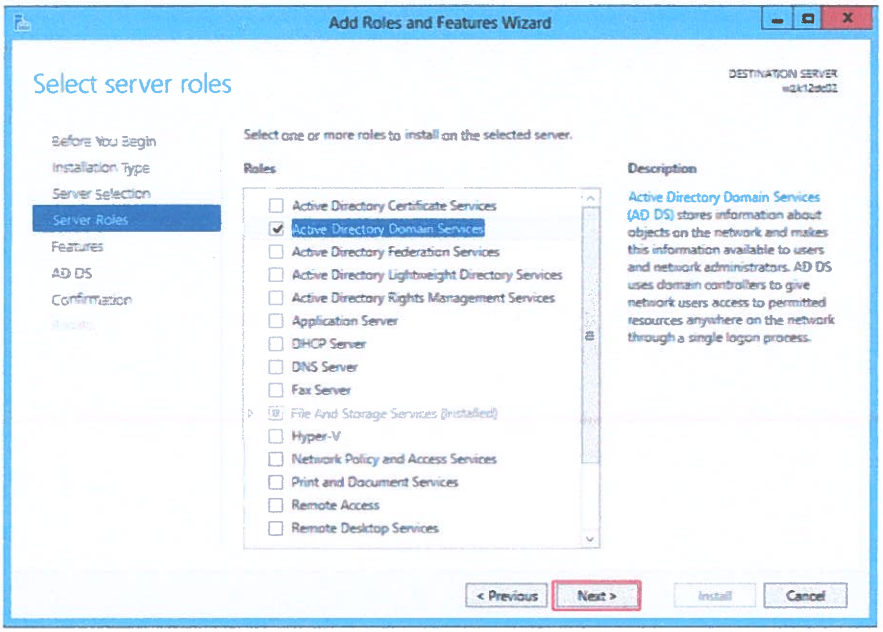


Figura 17: Active Directory Role.

Fonte: do autor.

No fim da instalação da *role* poderá ser necessário reiniciar o servidor para aplicar as definições. Após o arranque, irá surgir a opção do assistente de promoção de domínio e opta-se pela opção *Add a new forest* e dá-se o nome ao domínio PG.local. No passo seguinte, é necessário definir uma palavra-passe para o administrador de domínio. Neste caso definiu-se os seguintes acessos Administrator/Password01. Posteriormente será necessário reiniciar para aplicar as definições feitas.

Logo após a configuração da ADDS já é possível criar os utilizadores que se pretende, para o cliente1 que terá mais tarde permissões para autenticar na base de dados.

Dentro do *Active Directory* carrega-se em cima do domínio com o lado direito do rato em PG.local e escolhe-se *New User* (Figura 18) e define-se o Cliente1 para o nome de utilizador e Password01 para a password de acesso (Figura 19).

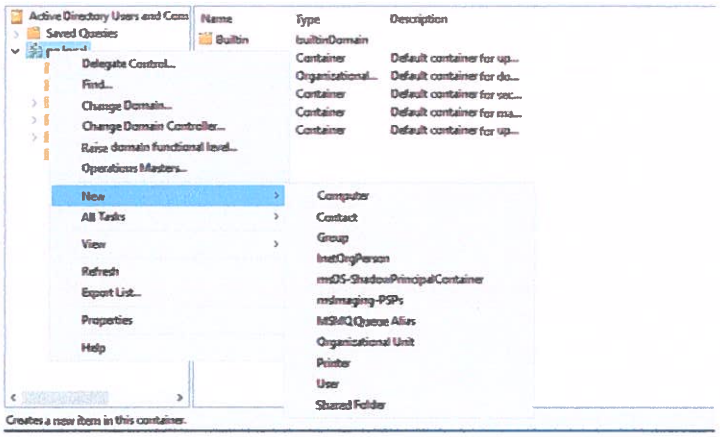


Figura 18: Create new user AD.

Fonte: do autor.

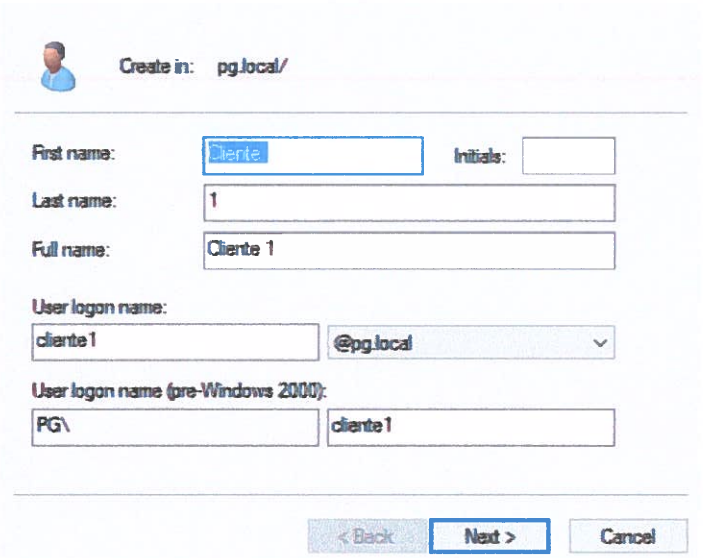


Figura 19: Create new user AD.

Fonte: do autor.

Configuração do *DHCP*

Para iniciar a configuração da role de DHCP será necessário abrir novamente o *Server Manager*, seleciona-se a opção *Add roles and features* e, em seguida, no separador *Server Roles* escolhe-se o DHCP (Figura 20).

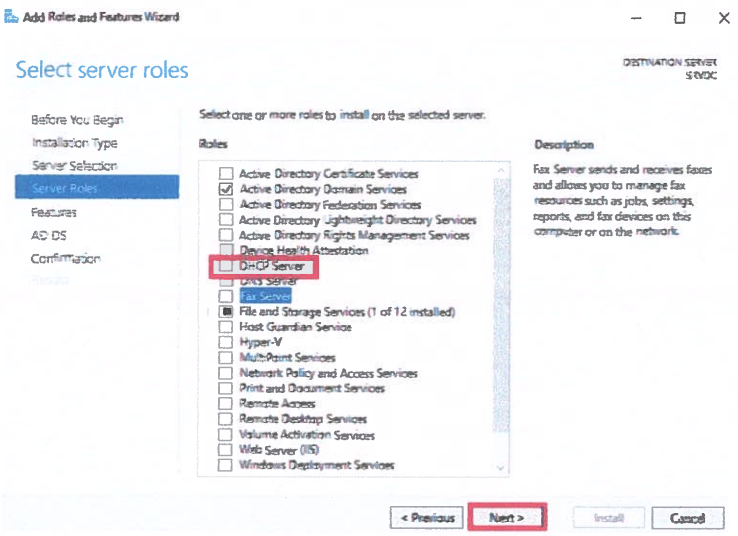


Figura 20: Instalar o DHCP.

Fonte: do autor.

Depois de finalizado o *wizard* da instalação será solicitada a configuração do DHCP, que vai permitir a criação de um *scope* de endereços de IP que irá ser atribuída pelas máquinas que se ligarem à rede. Dentro do DHCP existe uma ferramenta de gestão (DHCP *Manager*) que permitirá configurar a *range* de IPs, entre outros.

Para a configuração do DHCP define-se um *scope* de 20 endereços (192.168.2.10/24 -192.168.61.2.20/24) (Figura 21), o servidor de *DNS* será por sua vez o *domain controler* (192.168.2.1/24) (Figura 22) e, por fim, coloca-se como router o endereço 192.168.2.254 que pertence ao *Routing Server* (SRV-RSERVER) (Figura 23). Com as definições aplicadas no DHCP, todos os equipamentos que estiverem ligados na rede irão obter os parâmetros da rede indicados.

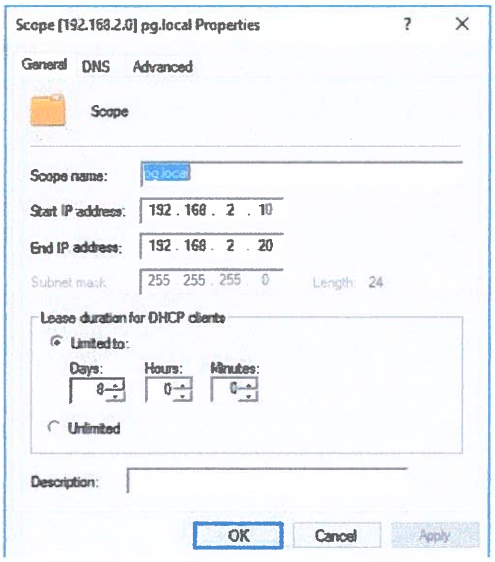


Figura 21: Scope DHCP.

Fonte: do autor.

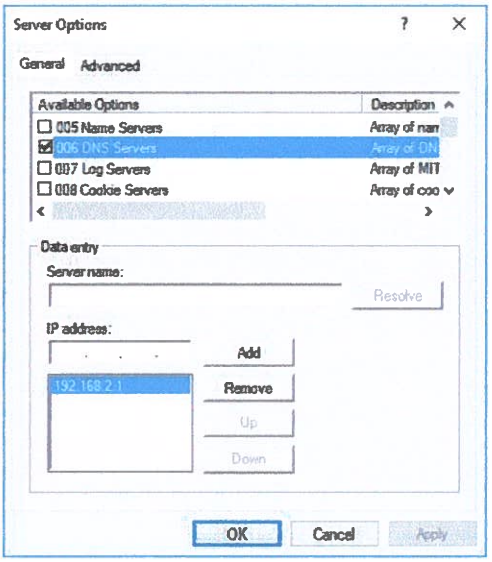


Figura 22: DNS no DHP.

Fonte: do autor.

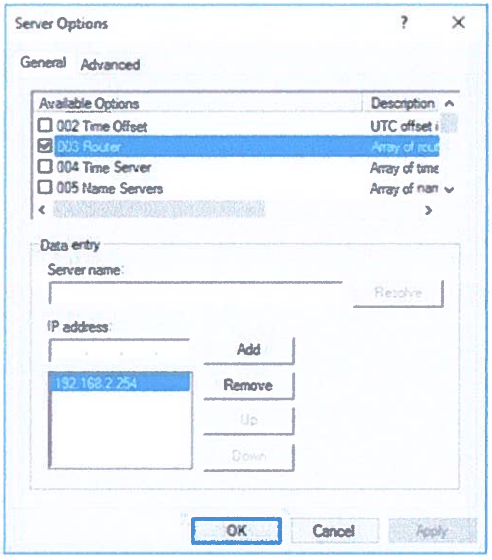


Figura 23: Router no DHCP.

Fonte: do autor.

Configuração do *DNS*

Instalar a role de DNS é um processo semelhante à role anterior. A instalação é feita a partir do *Server Manager* na opção adicionar ou remover *roles*. Deste modo, seleciona-se a opção DNS *Server* e com a instalação realizada pode-se então parametrizar uma zona primária que irá pertencer ao domínio PG.local e ao criar esta zona todos os pedidos de resolução de nome ou IP irão passar pelo servidor.

Configuração do *Network Policy Server*

Antes de efetuar a configuração do NPS é necessário instalar a role, para isso é necessário escolher a opção *Add Roles and Features* e a opção *Network Policy and Acess Services* dentro do Server Manager. De seguida, na página dos serviços da role, há que selecionar *Network Policy Server* e seguir a instalação. Com a instalação concluída, fica disponível nas ferramentas administrativas a consola de gestão.

Após toda a configuração da role, é possível escolher dentro do NPS a opção NAP (*Network Acess Protection*) para definir os requisitos necessários para o cliente aceder a rede. Portanto, para ativar esta opção será essencial escolher no separador do NPS a opção

configure NAP. Na configuração do *Network Acess Protection* escolhe-se a opção *Dynamic Host Configuration Protocol* (DHCP) e seleciona-se *Radius Cliente* que irá efetuar a verificação de todos os requisitos (Figura 24). De seguida, o próximo passo será selecionar o scope DHCP onde atuará a verificação de proteção, na janela seguinte é definida a página que reencaminhará o utilizador a corrigir as anomalias. Finalmente, para que se possa ver o funcionamento coloca-se o seguinte *site* http://istec.pt, pois este redireciona o utilizador em caso de incumprimento dos requisitos de rede.

Após a parametrização do NAP é associada uma política ao grupo dos *Domain Users* e para ver o funcionamento basta desligar a *firewall* no computador cliente (Figura 25).

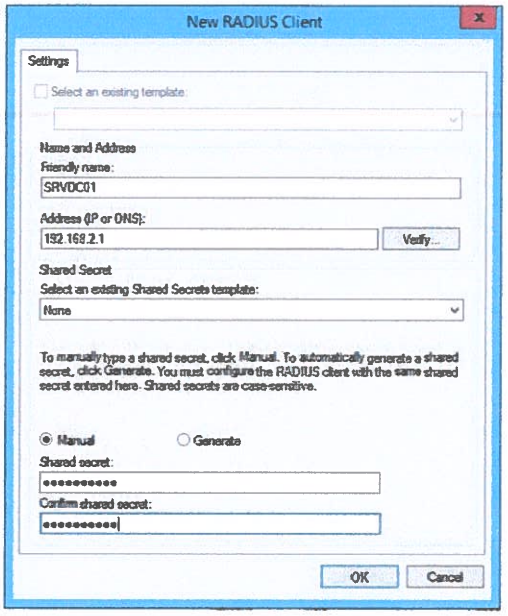


Figura 24: NAP Access test.
Fonte: do autor.

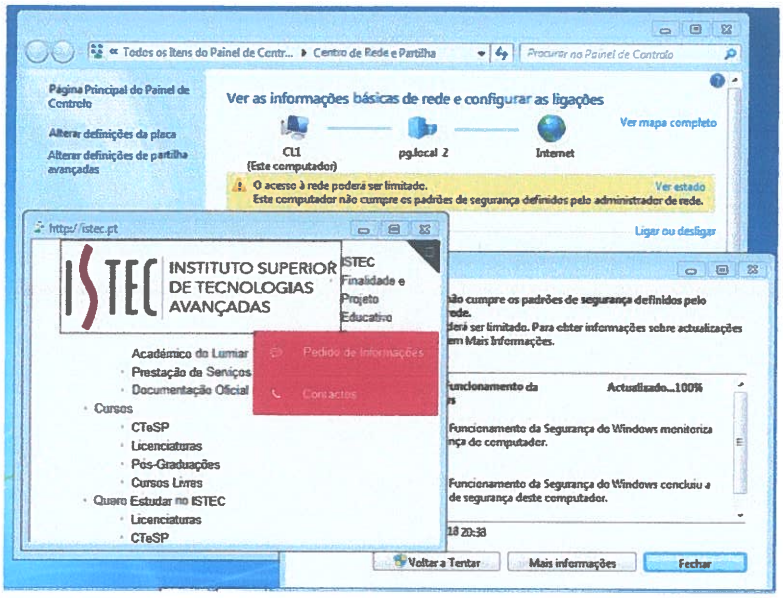


Figura 25: Teste no Cliente, serviço NPS.

Fonte: do autor.

Configuração *do Routing Server* (SRV-RSERVER)

Neste tipo de infraestrutura em que apenas existe uma comunicação para o exterior, vão ser utilizadas as duas placas de rede criadas anteriormente (*Internal* e *External*). Terminada a instalação do sistema operativo é necessário renomear o servidor para SVR-RSERVER e em seguida configurar a placa de rede com as seguintes informações:

IP: 192.168.2.254

Máscara: 255.255.255.0

Gateway: 192.168.2.2

DNS: 192.168.2.1

Depois de configurado será possível dentro da rede interna ter este servidor a servir de *router*. Para se poder agregar este servidor ao domínio é necessário seguir os seguintes passos: *System Properties* > *Computer Name* > *Change* > *Domain* > Escrever PG.local > *OK*. Consequentemente, é necessário reiniciar uma vez mais o servidor para

aplicar as definições. No seguimento da instalação da role de *routing server*, seleciona-se dentro do *System Manager* a opção *Add new role* e a opção *Remote Acess* (Figura 26).

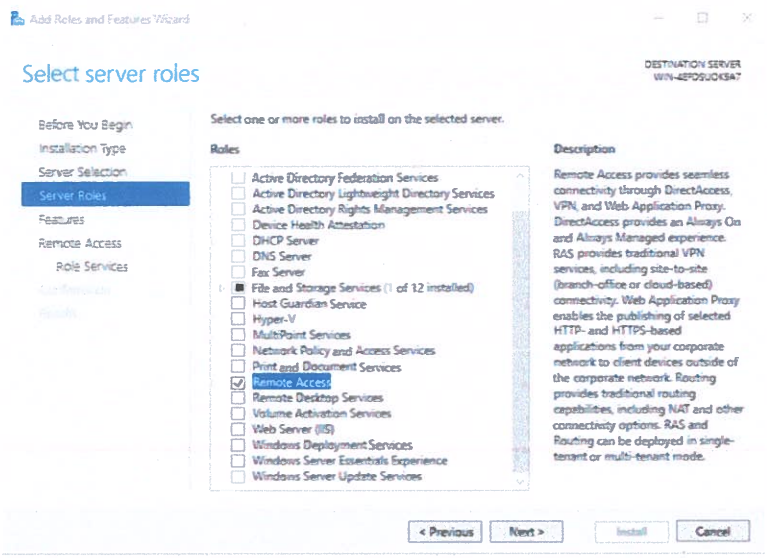


Figura 26: Remote Acess Role.

Fonte: do autor.

No separador *Role Services* é necessário escolher apenas a tecnologia de *routing* que irá permitir fazer a ligação entre as duas placas de rede (Figura 27).

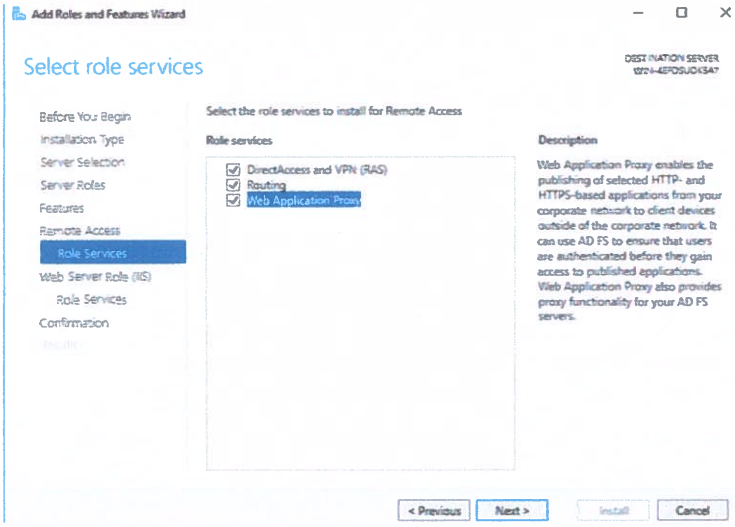


Figura 27: Routing.

Fonte: do autor.

Depois da instalação dos serviços de *Routing Server* procede-se à sua configuração nas ferramentas administrativas. Ao iniciar a consola seleciona-se a opção de NAT, que permite unir as duas placas de rede e define-se qual a placa de rede que irá

comunicar com o exterior (*External Network Project*), sendo a única saída para a *internet*. Findada a configuração do *Routing Server* é necessário garantir que o serviço está em funcionamento, sendo possível utilizar como *gateway* o IP que se definiu anteriormente (192.168.2.254). Para efetuar o teste de ligação basta tentar aceder a *internet* a partir das definições de rede informadas anteriormente (Figura 28).

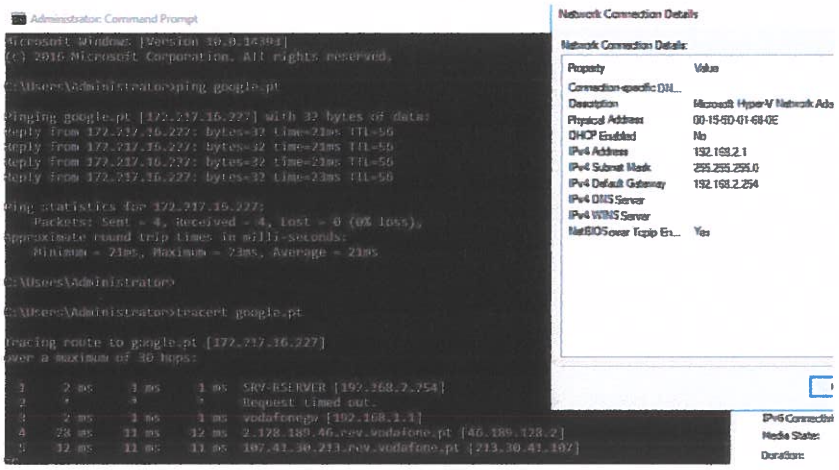


Figura 28: Routing Test.

Fonte: do autor.

Configuração do *SQL SERVER* (SRVSQL)

Para a configuração do servidor de SQL é necessário seguir os seguintes passos:

Nome do Servidor: SRVSQL

Configuração *IPv4*:

IP: 192.168.2.4

Máscara: 255.255.255.0

Gateway: 192.168.2.254

DNS: 192.168.2.1

Domínio: PG.local

Descativar *IPv6*.

Na *Firewall* é necessário ativar os Serviços de *ICMP*.

Concretizada a instalação e configuração anterior da máquina virtual, dá-se início à instalação: do SQL *Server 2016 Service Pack 1 Express* por meio do seguinte link

https://www.microsoft.com/en-us/download/confirmation.aspx?id=54284; e do *Microsoft* SQL *Management Studio 2017* (*SSMS*) acedendo a https://go.microsoft.com/fwlink/?linkid=873126.

Durante a instalação do SQL *Server* define-se uma instância com o nome SRVSQL\SQLSERVERPG, as credenciais do utilizador SA e com autenticação mista para se poder utilizar *Windows Athentication*. Após configurado o SQL *Server 2016* pode-se iniciar a instalação do *SSMS* que irá permitir a gestão da base de dados. No SQL *Manager* define-se o nome da instância, procede-se ao *login* com *Windows Authentication* e na opção de *Security* define-se um novo *login* (Cliente1) que pertence ao domínio, sendo também necessário dar permissões de escrita na base de dados. (Figura 29 e 30).

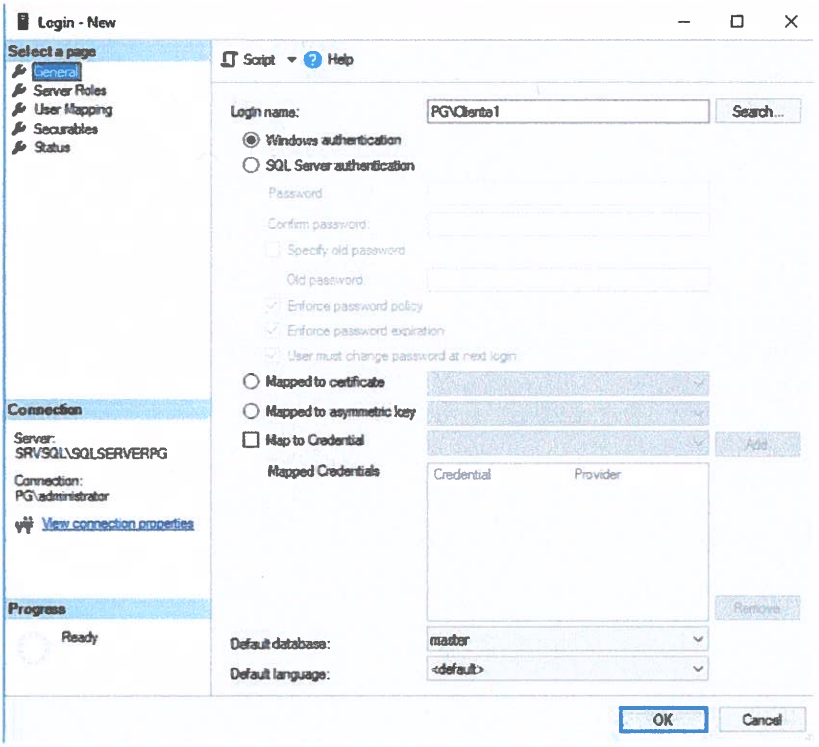


Figura 29: Acessos do utilizador Cliente1 na Base de dados.

Fonte: do autor.

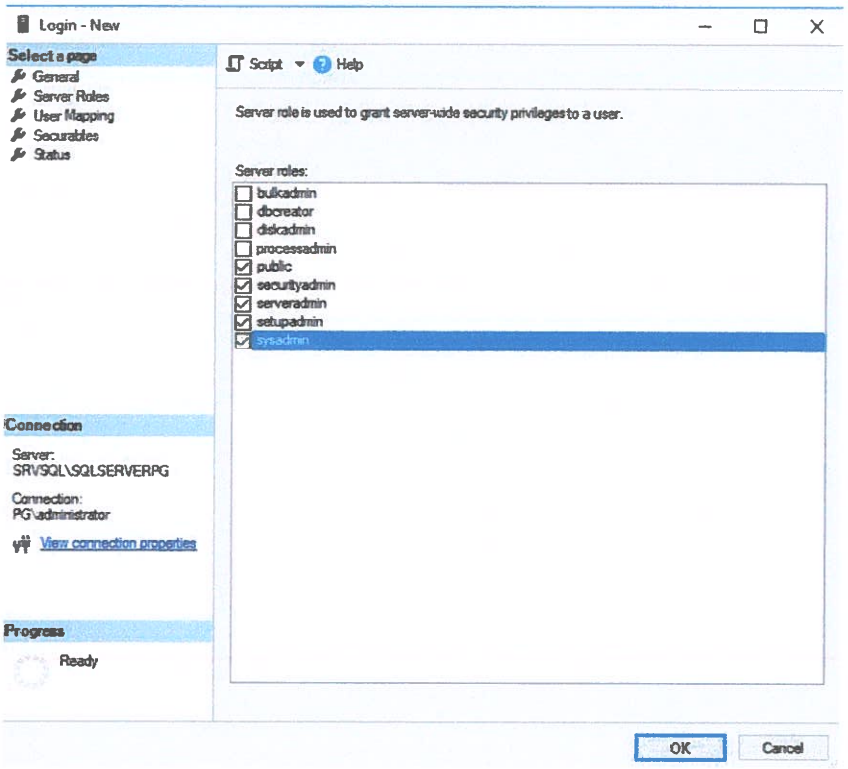


Figura 30: Permissões do Utilizador.

Fonte: do autor.

Depois da configuração das permissões do utilizador de domínio procede-se agora à construção das bases de dados, carrega-se em *New Query* dentro da instância e coloca-se a seguinte informação (Figura 31):

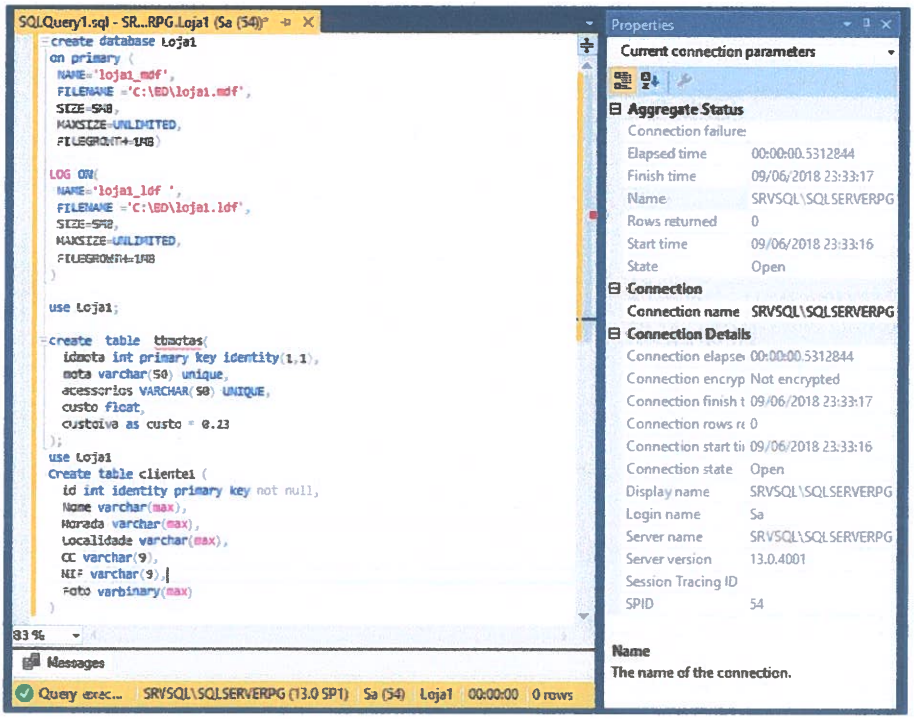


Figura 31: Query da construção das bases de dados.

Fonte: do autor.

Configuração do *Cliente*

Para a configuração da máquina cliente é necessário seguir os seguintes passos:

Nome da Máquina: CL1

Configuração *Rede*: DHCP

Será atribuído um IP entre 192.168.2.10 e 192.168.2.20.

Domínio: PG.local

O processo de instalação da máquina cliente é semelhante ao dos servidores configurados anteriormente, porem a imagem do sistema operativo utilizada será diferente (Windows 7 Pro x86). Após instalada a máquina virtual dá-se agora inicio à parametrização para que se possa programar aplicação e publica-la para isso é necessário instalar o *Visual Studio* e o SSMS, utilizando os links anteriores de downloads, será também necessário instalar o Adobe Reader para ler os documentos pdfs provenientes da aplicação. Toda a configuração deverá ser feita já dentro do utilizador de domínio (Cliente1).

ADO.NET

Esta fase concerne no desenvolvimento da aplicação em *ADO.net*. Assim, dentro do *Visual Studio* é necessário fazer novo projeto, escolher a ferramenta WPF e, posteriormente, dá-se o nome do projeto POS (*Point of Sale)* Barrigas do Asfalto. A elaboração deste programa tem como objetivo um ponto de venda para lojas de motas logo, a autenticação passa por utilizadores autenticados no domínio onde existe comunicação com o SRVSQL que contém a base de dados com as informações das motas, preço, acessórios e clientes. Este ponto de venda permite ainda a edição das tabelas das motas, outros acessórios e dos preços sendo possível editar valores antigos e até mesmo elimina-los. Em todos os separadores da aplicação será possível ver o que está no carrinho de compras e finalizar os pedidos. Será também adicionado uma calculadora que permitirá fazer contas em caso de dúvidas, abaixo encontra-se algumas fotos do que será elaborado.

Janela de Autenticação

Para a elaboração da página de autenticação é necessário definir quais os dados a utilizar para o *login*, neste caso foi utilizado nome de utilizador e *password*, sendo a autenticação feita a partir do utilizador de domínio. Seguidamente, na *MainWindow.*XML define-se o tamanho da janela, o título da janela e, também, é colocada a opção para não redimensionar a janela (*NoResize*) (Figura 32).

Title="POS-Barrigas Do Asfalto" Height="278" Width="353"
ResizeMode="NoResize">

```
<Window x:Class="WpfApp1.MainWindow"
        xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
        xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
        xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
        xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
        xmlns:local="clr-namespace:WpfApp1" mc:Ignorable="d"
        Title="POS-Barrigas Do Asfalto" Height="278" Width="353" ResizeMode="NoResize"
Icon="C:\Barrigas 2\wpfapp1\WpfApp1\Imagens\ico.png">
```

Figura 32: MainWindow de Autenticação.

Fonte: do autor.

Além disso, é adicionada uma *Grid* onde serão adicionados os botões *Login* e Sair, uma *Textbox*, uma *PasswordBox* para ocultar a palavra-passe introduzida, as *labels* para dar informação ao utilizador do que introduzir nas caixas de texto e, por fim, as imagens (Figura 33). Nos Botões *Login* e Sair é necessário associar o *Click*="" para criar o evento no *code behind*, sendo que o funcionamento desta janela será descrito posteriormente.

```
    <Grid Margin="-74,0,-70,-270">
        <TextBox x:Name="txtuser"  HorizontalAlignment="Left" Height="23"
Margin="202,108,0,0" TextWrapping="Wrap" Text="" VerticalAlignment="Top" Width="155"/>
        <PasswordBox x:Name="txtpassword" HorizontalAlignment="Left" Margin="202,162,0,0"
VerticalAlignment="Top" Width="155" Height="22"/>
        <Label Content="Utilizador:" HorizontalAlignment="Left" Margin="200,77,0,0"
VerticalAlignment="Top" RenderTransformOrigin="0.508,1.071"/>
        <Label Content="Password:" HorizontalAlignment="Left" Margin="201,136,0,0"
VerticalAlignment="Top" RenderTransformOrigin="1.319,0.351"/>
        <Button x:Name="Btlogin" Content="Login" HorizontalAlignment="Left"
Margin="202,189,0,0" VerticalAlignment="Top" Width="75" Click="Btlogin_Click"
RenderTransformOrigin="0.467,-0.15"/>
        <Button Content="Sair" HorizontalAlignment="Left" Margin="282,189,0,0"
VerticalAlignment="Top" Width="75" Click="Button_Click"/>

        <Image HorizontalAlignment="Left" Height="100" Margin="96,93,0,0"
VerticalAlignment="Top" Width="100" Source=".\Imagens\lock.png"/>
        <Image HorizontalAlignment="Left" Height="78" Margin="113,10,0,0"
VerticalAlignment="Top" Width="274" RenderTransformOrigin="0.36,0.25"
Source=".\Imagens\logo4.jpg"/>
    </Grid>
```

Figura 33: MainWindow Login

Fonte: do autor.

No *code behind* da página *MainWindow* pode-se então dar início à configuração da autenticação do utilizador e para tal é necessário utilizar uma *DLL* do Windows que guarde as informações dos utilizadores autenticados. Portanto, todos os utilizadores que estejam criados no domínio e que façam login na máquina terão acesso à aplicação. Primeiramente, é necessário importar a *DLL* para o código (*DllImport("advapi32.dll")*) *e,* em seguida é criada uma função que irá receber o nome, o domínio, e o tipo de login devolvendo um booleano. O objetivo é ler o utilizador e palavra passe recebidas e comprar os valores, permitindo o acesso ao interior do programa, sendo o processo efetuado dentro do evento do botão de login (Figura 33).

```
namespace WpfApp1
{    public partial class MainWindow : Window
    {
        public MainWindow()
        {InitializeComponent();        }

        [DllImport("advapi32.dll")]
        public static extern bool LogonUser(string name, string domain, string pass, int
logType, int logpv, ref IntPtr pht);
        private void Btlogin_Click(object sender, RoutedEventArgs e)
        {
            IntPtr th = IntPtr.Zero;
            bool log = LogonUser(txtuser.Text, "pg.local", txtpassword.Password, 2, 0, ref
th);
            if (log)
            {
                this.Hide();
                Menus ss = new Menus();
                ss.Show();}
```

Figura 34: MainWindows Login, Code behind.

Fonte: do autor.

Utiliza-se dentro do evento a opção *This.Hide*() para ocultar a janelas de *login* em caso de sucesso e *ss.Show()* para mostrar os menus. No botão de sair o procedimento é semelhante, utiliza-se *this.Close()* para fechar (Figura 35). Na Figura 36 é possível ver o design final da página.

```
      private void Button_Click(object sender, RoutedEventArgs e)
        {
            this.Close();
        }
```

Figura 35: Botão de Sair.

Fonte: do autor.

Figura 36: Janela de Autenticação.

Fonte: do autor.

Janela de vendas

Para criar uma nova janela é necessário carregar em cima do nome projeto com o lado direito do rato e escolher as opções adicionar e nova janela, e dá-se o nome de Menus. Já dentro do ficheiro Menus.xml cria-se a *grid*, sendo que a mesma irá suportar as seguintes *tabs*: Motas, Acessórios, Carrinho, Pagamento. Para cada *separador* inicia-se sempre da seguinte forma:

```
<TabControl>
    <TabItem Header="Motas"> (Conteúdo da Janela) </TabItem>
    <TabItem Header="Acessórios"> (Conteúdo da Janela) </TabItem>
    <TabItem Header="Carrinho"> (Conteúdo da Janela) </TabItem>
    <TabItem Header="Pagamento"> (Conteúdo da Janela) </TabItem>
</TabControl>
```

Relativamente à primeira, *tab* (Motas) será constituída sete *labels* que irão conter as informações sobre os botões que serão adicionados. Neste caso serão colocados seis botões que vão conter as imagens das motas e mais três botões para finalizar, aceder aos acessórios ou fazer *logout* da aplicação. Para os botões adicionados será sempre necessário associar estes a um evento para mais tarde serem geridos (Figura 37). Na construção dos quatro separadores será necessário adicionar quatro *DataGrids* diferentes

que irão permitir visualizar o cesto das compras, fazendo o *Binding* dos valores que é pretendido (Figura 38). Na Figura 39 pode ser analisado o *design* da parte xml.

Figura 37: Construção dos Menus (Motas).

Fonte: do autor.

```
      <DataGrid Name="gridmotas"  HorizontalAlignment="Left" Height="239"Margin="372,10,0,0"
VerticalAlignment="Top" Width="161" >
        <DataGrid.Columns>
                <DataGridTextColumn Header="Mota" Binding="{Binding mota}" />
                <DataGridTextColumn Header="Acessorios" Binding="{Binding acessorios}" />
                <DataGridTextColumn Header="c/Iva" Binding="{Binding custoiva}"/>
         </DataGrid.Columns>
       </DataGrid>
```

Figura 38: DataGrid Motas.

Fonte: do autor.

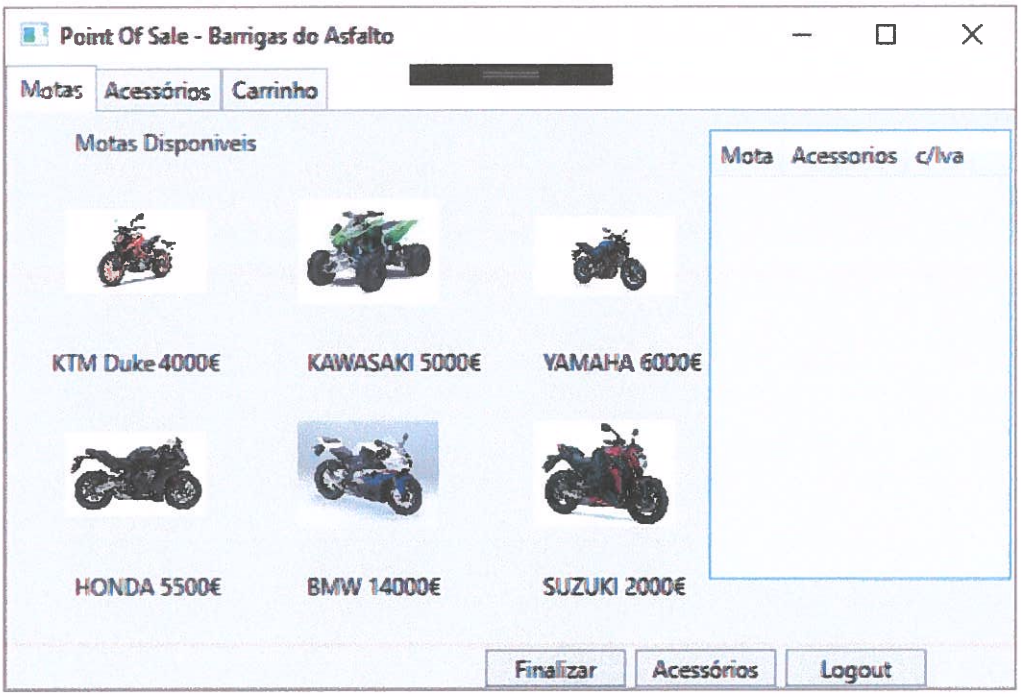


Figura 39: Separador das Motas.

Fonte: do autor.

Para o separador dos acessórios repete-se o processo descrito anteriormente, à exceção do botão acessórios que não é incluído, sendo necessário também a alteração das imagens nos seis (Figura 40, 41 e 42).

Figura 40: Construção dos Menus (Motas).

Fonte: do autor.

```
    <DataGrid Name="gridacessorios" Grid.Column="2" HorizontalAlignment="Left" Height="239"
Margin="347,11,0,0" VerticalAlignment="Top" Width="161">
        <DataGrid.Columns>
           <DataGridTextColumn Header="Mota" Binding="{Binding mota}" />
           <DataGridTextColumn Header="Acessorios" Binding="{Binding acessorios}" />
           <DataGridTextColumn Header="c/Iva" Binding="{Binding custoiva}"/>
                    </DataGrid.Columns>
                </DataGrid>
```

Figura 41: DataGrid Acessório.

Fonte: do autor.

Figura 42: Separador dos Acessórios.

Fonte: do autor.

Para a construção do separador carrinho, coloca-se uma *DataGrid* que irá receber os valores da base de dados, adiciona-se quatro *TextBoxs* que vão permitir introduzir dados como, por exemplo, o Idmota, a mota, os acessórios, o preço s/iva, acompanhados de *labels*. Acrescentam-se também quatro botões (inserir, atualizar, eliminar e dúvidas nas contas?) para permitir modificar os dados na tabela e aceder à calculadora (Figura 43). Contudo, é necessário que cada botão tenha o seu próprio evento (Figura 44) e, é adicionada uma *label* que irá receber o valor total dos *itens* que se encontram no carrinho de compras. Na Figura 45 é possível ver o *design* após finalizado a contrução do código XML.

```
      <DataGrid Name="grdListaTudo" CanUserAddRows="False" AutoGenerateColumns="False"
Margin="10,13,149,57" SelectionChanged="grdListaTudo_SelectionChanged">
        <DataGrid.Columns>
              <DataGridTextColumn Header="idmota" Binding="{Binding idmota}" />
              <DataGridTextColumn Header="Mota" Binding="{Binding mota}" />
              <DataGridTextColumn Header="Acessorios" Binding="{Binding acessorios}" />
              <DataGridTextColumn Header="Valor" Binding="{Binding custo}" />
              <DataGridTextColumn Header="c/Iva" Binding="{Binding custoiva}"/>
        </DataGrid.Columns>
```

Figura 43: DataGrid (Carrinho).

Fonte: do autor.

```
<Label Content="Total:" HorizontalAlignment="Left" Margin="386,279,0,0" VerticalAlignment="Top"/>
<TextBlock HorizontalAlignment="Left" Margin="440,283,0,0" VerticalAlignment="Top" Name="TotalPrice" RenderTransformOrigin="0.5,0.5" Width="78" Height="16"...>
<TextBox Name="tbmota" HorizontalAlignment="Left" Height="23" Margin="298,76,0,0" TextWrapping="Wrap" VerticalAlignment="Top" Width="120" />
<TextBox Name="tbacessorio" HorizontalAlignment="Left" Height="23" Margin="298,129,0,0" TextWrapping="Wrap" VerticalAlignment="Top" Width="120"/>
<TextBox Name="tbvalor" HorizontalAlignment="Left" Height="23" Margin="298,164,0,0" TextWrapping="Wrap" VerticalAlignment="Top" Width="120"/>
<Button Name="btinsert" Content="Insert" HorizontalAlignment="Left" Margin="416,152,0,0" VerticalAlignment="Top" Width="75" Click="btinsert_Click"/>
<Button Name="btupdate" Content="Update" HorizontalAlignment="Left" Margin="416,217,0,0" VerticalAlignment="Top" Width="75" Click="btupdate_Click"/>
<Button Name="btdelete" Content="Delete" HorizontalAlignment="Left" Margin="416,242,0,0" VerticalAlignment="Top" Width="75" RenderTransformOrigin="0.44,1.25" Click="btdelete_Click"/>
<Button x:Name="calculadora" Content="Duvida nas Contas? " HorizontalAlignment="Left" Margin="10,279,0,0" VerticalAlignment="Top" Width="129" Click="calculadora_Click" />
<TextBlock HorizontalAlignment="Left" Margin="442,55,0,0" TextWrapping="Wrap" Text="Mota" VerticalAlignment="Top" RenderTransformOrigin="1.5,0.563"/>
<TextBlock HorizontalAlignment="Left" Margin="438,99,0,0" TextWrapping="Wrap" Text="Acessórios" VerticalAlignment="Top" RenderTransformOrigin="0.571,1.812"/>
<TextBlock HorizontalAlignment="Left" Margin="428,143,0,0" TextWrapping="Wrap" Text="Preço s/Iva" VerticalAlignment="Top"/>
<TextBlock HorizontalAlignment="Left" Margin="442,13,0,0" TextWrapping="Wrap" VerticalAlignment="Top" RenderTransformOrigin="0.383,0.581"><Run Text="ID"/><Run Text="Mota"/></TextBlock>
<TextBox x:Name="tbidmota" HorizontalAlignment="Left" Height="23" Margin="358,32,0,0" TextWrapping="Wrap" VerticalAlignment="Top" Width="120" />
```

Figura 44: Construção dos Menus.

Fonte: do autor.

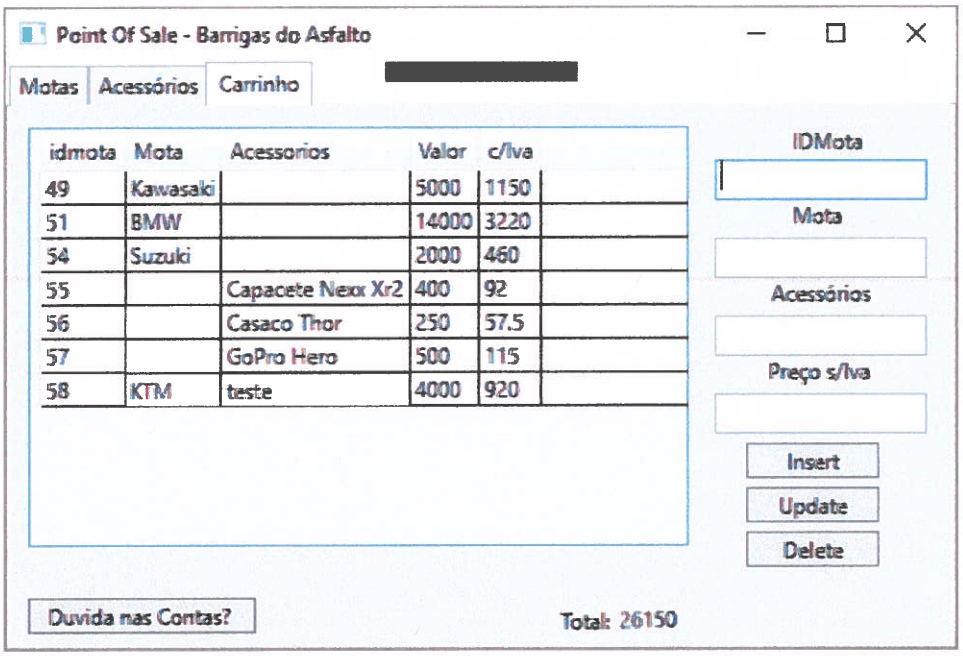


Figura 45: Separador do Carrinho.

Fonte: do autor.

No que concerne à construção do separador pagamentos há que adicionar os seguintes *itens* no XML: uma *DataGrid* para mostrar os valores que estão na base de dados, uma imagem, seis TextBox's para introduzir dados, *labels* e os botões *Delete*, *PDF* e Procurar, Salvar Dados. Para cada botão é necessário definir o evento a partir do *Click*="" (Figura 46 e 47).

```
<TabItem x:Name="tabpagamento" Header="Pagamento" HorizontalAlignment="Right" Margin="0" RenderTransformOrigin="0.556,1" Width="72" Height="25" VerticalAlignment="Bottom">
    <Grid Background="#FFE5E5E5" Margin="-4,0,-6,0">
        <TextBox Name="Tbnome" HorizontalAlignment="Left" Height="19" Margin="24,52,0,0" TextWrapping="Wrap" VerticalAlignment="Top" Width="104" TextChanged="TextBox_TextChanged_1"/>
        <TextBox Name="tbcc" HorizontalAlignment="Left" Height="19" Margin="24,96,0,0" TextWrapping="Wrap" VerticalAlignment="Top" Width="104" RenderTransformOrigin="0.481,1.211" TextChanged="tbcc_TextChanged"/>
        <TextBox Name="tbmorada" HorizontalAlignment="Left" Height="19" Margin="173,34,0,0" TextWrapping="Wrap" VerticalAlignment="Top" Width="155"/>
        <TextBox Name="tblocalidade" HorizontalAlignment="Left" Height="19" Margin="24,182,0,0" TextWrapping="Wrap" VerticalAlignment="Top" Width="104"/>
        <TextBox Name="tbnif" HorizontalAlignment="Left" Height="19" Margin="24,137,0,0" TextWrapping="Wrap" VerticalAlignment="Top" Width="104" TextChanged="TextBox_TextChanged" RenderTransformOrigin="0.433,0.105"/>
        <TextBlock HorizontalAlignment="Left" Margin="29,33,0,0" TextWrapping="Wrap" Text="Nome:" VerticalAlignment="Top" RenderTransformOrigin="1.618,0.313"/>
        <TextBlock HorizontalAlignment="Left" Margin="29,76,0,0" TextWrapping="Wrap" Text="NºCC:" VerticalAlignment="Top"/>
        <TextBlock HorizontalAlignment="Left" Margin="172,13,0,0" TextWrapping="Wrap" VerticalAlignment="Top" RenderTransformOrigin="0.614,-0.312"><Run Text="Morada"/><Run Text=":"/></TextBlock>
        <TextBlock HorizontalAlignment="Left" Margin="29,186,0,0" TextWrapping="Wrap" VerticalAlignment="Top"><Run Text="Localidade"/><Run Text=":"/></TextBlock>
        <TextBlock HorizontalAlignment="Left" Margin="29,121,0,0" TextWrapping="Wrap" Text="NIF:" VerticalAlignment="Top"/>
        <TextBlock HorizontalAlignment="Left" Margin="157,63,0,0" TextWrapping="Wrap" VerticalAlignment="Top" Height="28" Width="88"><Run Text="Foto de Perfil:"/><LineBreak/><Run/></TextBlock>
        <Button Click="Procurar_Click" Content="Procurar" HorizontalAlignment="Left" Margin="247,63,0,0" VerticalAlignment="Top" Width="52"/>
        <Label Content="" HorizontalAlignment="Left" Margin="157,124,0,0" VerticalAlignment="Top" RenderTransformOrigin="0.185,0.346" Width="171"/>
        <Image Name="imgem" HorizontalAlignment="Left" Height="108" Margin="299,18,0,0" VerticalAlignment="Top" Width="103"/>
        <Button Click="PDF_Click" Content="PDF" HorizontalAlignment="Left" Margin="375,265,0,0" VerticalAlignment="Top" Width="75" RenderTransformOrigin="0.72,0.5"/>
        <Button Click="EXCEL_Click" Content="EXCEL" HorizontalAlignment="Left" Margin="453,264,0,0" VerticalAlignment="Top" Width="75"/>
        <Button Content="Salvar dados" HorizontalAlignment="Left" Margin="29,286,0,0" VerticalAlignment="Top" Width="75" Click="Salvardados_Click"/>
        <DataGrid Name="gridpagamento" HorizontalAlignment="Left" Height="131" Margin="257,125,0,0" VerticalAlignment="Top" Width="303" SelectionChanged="DataGrid_SelectionChanged"...>
        <TextBlock HorizontalAlignment="Left" Margin="289,264,0,0" TextWrapping="Wrap" VerticalAlignment="Top" Width="71" Text="Emitir fatura:"/>
        <TextBlock HorizontalAlignment="Left" Margin="24,18,0,0" TextWrapping="Wrap" VerticalAlignment="Top" Text="Dados para pagamento"/>
        <TextBox Name="tbdelete" HorizontalAlignment="Left" Height="23" Margin="11,267,0,0" TextWrapping="Wrap" VerticalAlignment="Top" Width="29" TextChanged="TextBox_TextChanged_2"/>
```

Figura 46: Construção de Menus (Acessórios).

Fonte: do autor.

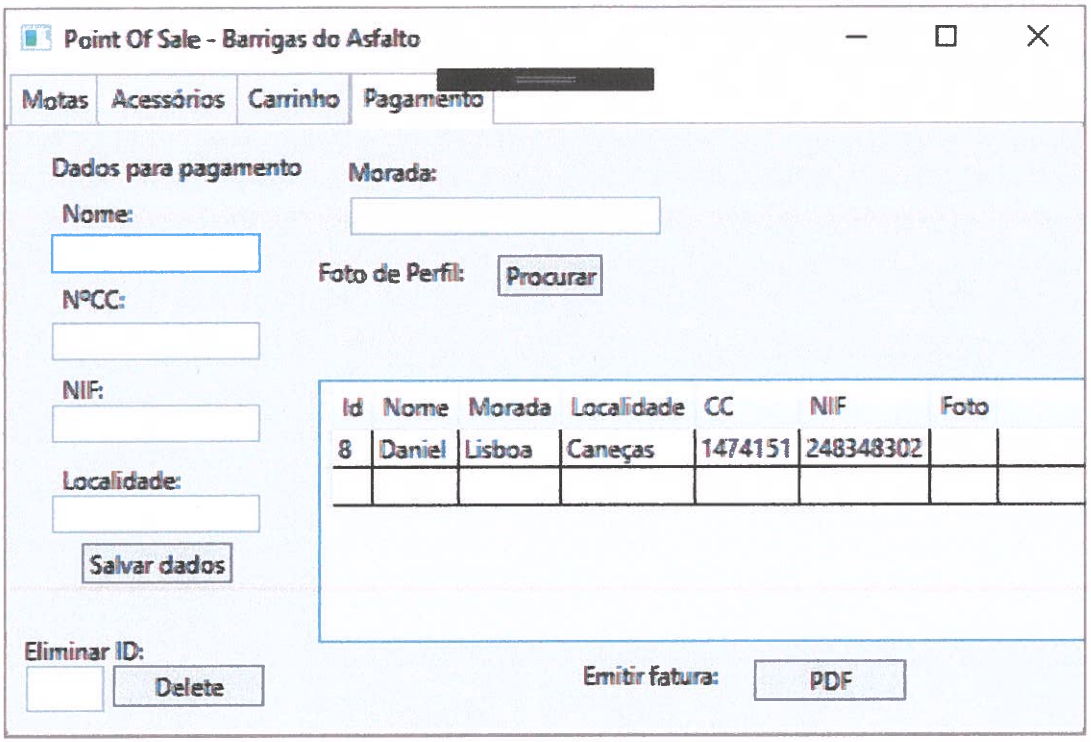


Figura 47: Separador Pagamento.

Fonte: do autor.

No *code behind* da página *Menus* procede-se à configuração dos botões e à sua integração com a base de dados. É necessário criar uma classe que recebe os valores (nome, acessórios e custo) que são provenientes dos botões que foram criados (Figura 48).

```
public class veiculo
  {
      public string nome { get; set; }
      public string acessorios { get; set; }
      public int custo { get; set; }
      //public int moeda { get; set; }
  }
```

Figura 48: Classe Veiculo.

Fonte: do autor.

Dentro do evento do botão finalizar é necessário que o mesmo reencaminhe o utilizador para o separador final (carrinho) por meio de finalizar a compra. Para isso será necessário escrever o nome da *tab* (tabcarrinho) e em seguida utilizar *.iselected = true* para redirecionar para a separador. Este mesmo processo é utilizador para o botão de *logout* (Figura 49).

```
private void Button_logout(object sender, RoutedEventArgs e)
{
    this.Close();
    MainWindow mm = new MainWindow();
    mm.Show();
}
private void Pagar_click(object sender, RoutedEventArgs e)
{
    tabpagamento.IsSelected = true;
}
private void Button_Click_1(object sender, RoutedEventArgs e)
{
    tabcarrinho.IsSelected = true;
}
private void Button_Click_2(object sender, RoutedEventArgs e)
{
    tabacessorios.IsSelected = true;
}
private void Button_Click_3(object sender, RoutedEventArgs e)
{
    tabcarrinho.IsSelected = true;
}
private void calculadora_Click(object sender, RoutedEventArgs e)
{
    Calculadora ss1 = new Calculadora();
    ss1.Show();
}
protected void Motas_Event(object sender, EventArgs args)
{
    Button MotasEvent = sender as Button;

}
```

Figura 49: Eventos tab (logout/Finalizar/Alternar).

Fonte: do autor.

Antes da atribuição dos valores aos botões das motas e acessórios é necessário criar uma função que permita pegar nos valores atribuídos e que os insira na base de dados, a mesma função terá o nome de actualiza e terá, também, o objectivo de preencher as *DataGrid* de ambos separadores (Figura 50).

```
void actualiza(int custo, string nome, string acessorios)
{
    using (Loja1Entities6 db = new Loja1Entities6())
    {
        tbmota mymota = new tbmota
        {
            mota = nome,
            acessorios = acessorios,
            custo = custo,
        };
        db.tbmotas.Add(mymota);
        db.SaveChanges();
        double? valorTotal = db.tbmotas.Sum(s => s.custo);
        totalPrice.Text = valorTotal.ToString();
        grdListaTudo.ItemsSource = db.tbmotas.ToList();
        gridmotas.ItemsSource = db.tbmotas.ToList();
        gridacessorios.ItemsSource = db.tbmotas.ToList();
    }
}
```

Figura 50: Função para Atualizar.

Fonte: do autor.

A figura 51 apresenta o código que permite fazer a soma da tabela de custos que será apresentada na *label* que se encontra no separador do carrinho, este excerto de código será aplicado em todos os eventos que interajam com a base de dados. O próximo passo será adicionar os valores aos botões das motas e acessórios pretendidos tendo em conta a classe do veículo que foi criada. No caso das motas não adicionamos os acessórios e no botão dos acessórios não se adiciona as motas, visto serem produtos independentes (Figura 52).

```
double? valorTotal = db.tbmotas.Sum(s => s.custo);
```

Figura 51: Soma da tabela Custo.

Fonte: do autor.

```
1 referência
private void KTM_Click(object sender, RoutedEventArgs e)
{
    actualiza(4000, "KTM", "");
}
1 referência
private void Kawasaki_Click(object sender, RoutedEventArgs e)
{
    actualiza(5000, "Kawasaki", "");
}
1 referência
private void BMW_Click(object sender, RoutedEventArgs e)
{
    actualiza(10000, "BMW", "");
}
1 referência
private void Suzuki_Click(object sender, RoutedEventArgs e)
{
    actualiza(2000, "Suzuki", "");
}
1 referência
private void Yamaha_Click(object sender, RoutedEventArgs e)...
1 referência
private void Honda_Click(object sender, RoutedEventArgs e)...
1 referência
private void Button_luvas(object sender, RoutedEventArgs e)
{
    actualiza(120, "", "Luvas Drenaline");
}
1 referência
private void Button_Capacete(object sender, RoutedEventArgs e)...
1 referência
private void Button_botas(object sender, RoutedEventArgs e)...
1 referência
private void Button_Casaco(object sender, RoutedEventArgs e)...
1 referência
private void Button_Camara(object sender, RoutedEventArgs e)...
1 referência
private void Button_viseira(object sender, RoutedEventArgs e)...
```

Figura 52: Eventos dos botões (Motas e Acessórios).

Fonte: do autor.

Após a colocação de todos os valores nos eventos das motas e acessórios é necessário configurar os botões para inserir, eliminar e actualizar. Para o evento do botão *Insert* é necessário passar os valores que serão escritos nas caixas de texto e passar para a base de dados a partir de Enteties instanciamos mymota que irá permitir passar os

valores a partir das variáveis (mota, acessórios, e custo), efectua-se *tryparse* para o caso de os campos estarem vazios (Figura 53).

```
        private void btinsert_Click(object sender, RoutedEventArgs e)
        {
            using (Loja1Entities6 db = new Loja1Entities6())
            {
                if (!int.TryParse(tbvalor.Text, out int valor))
                {
                    valor = 0;
                }
                tbmota mymota = new tbmota
                {
                    mota = tbmota.Text,
                    acessorios = tbacessorio.Text,
                    custo = valor
                };
                db.tbmotas.Add(mymota);
                db.SaveChanges();

                double? valorTotal = db.tbmotas.Sum(s => s.custo);
                totalPrice.Text = valorTotal.ToString();

                grdListaTudo.ItemsSource = db.tbmotas.ToList();
```

Figura 53: Função para inserir.

Fonte: do autor.

Para o evento actualizar o procedimento é semelhante ao anterior porém será necessário utilizar o valore do ID da mota para depois ser feita a actualização dos campos na base de dados. É criada uma variável ID Mota para que seja guardado o valor que é introduzido na caixa de texto ID Mota, se o valor corresponder ao que se encontra já entroduzido na lista então será permitido actualizar os valores das restantes caixas de texto (Figura 54).

```
    private void btupdate_Click(object sender, RoutedEventArgs e)
        {
            using (Loja1Entities6 db = new Loja1Entities6())
            {
                int idmota;
                if (int.TryParse(tbidmota.Text, out idmota))
                {
                    tbmota mymota = db.tbmotas.Where(x => x.idmota ==
idmota).FirstOrDefault();
                    db.SaveChanges();
                    mymota.mota = tbmota.Text;
                    mymota.acessorios = tbacessorio.Text;
                    mymota.custo = double.Parse(tbvalor.Text);

                    double? valorTotal = db.tbmotas.Sum(s => s.custo);
                    totalPrice.Text = valorTotal.ToString();

                    grdListaTudo.ItemsSource = db.tbmotas.ToList();
                }
            }
        }
```

Figura 54: Botão para atualizar.

Fonte: do autor.

No evento do botão eliminar os registos da base de dados o procedimento é semelhante ao acima descrito, contudo é necessário criar uma variável para receber o ID da mota e compará-lo com o registo da base de dados e caso exista elimina-o, se o utilizador tentar eliminar um registo que já tenha sido apagado será mostrada uma mensagem ao utilizador (Figura 55).

```
private void btdelete_Click(object sender, RoutedEventArgs e)
        {
            using (Loja1Entities6 db = new Loja1Entities6())
            {
                int idmota;
                if (int.TryParse(tbidmota.Text, out idmota))
                {
                 tbmota mymota = db.tbmotas.Where(x => x.idmota ==
idmota).FirstOrDefault();
                    if (mymota != null) db.tbmotas.Remove(mymota);
                    else MessageBox.Show("Este registo já foi eliminado!");
                    db.SaveChanges();
                    double? valorTotal = db.tbmotas.Sum(s => s.custo);


                    totalPrice.Text = valorTotal.ToString();
                    grdListaTudo.ItemsSource = db.tbmotas.ToList();
                }
            }
        }
```

Figura 55: Botão para eliminar.

Fonte: do autor.

No evento botão de dúvidas nas contas será efetuado um redireccionamento para uma nova página em XML que será uma calculadora (Figura 56).

```
        private void calculadora_Click(object sender, RoutedEventArgs e)
        {
            Calculadora ss1 = new Calculadora();
            ss1.Show();
        }
```

Figura 56: Chamar a página da Calculadora.

Fonte: do autor.

Para criar uma nova página é necessário carregar em cima do projeto e adicionar nova página XML e dá-se o nome de Calculadora. Para elaboração desta calculadora serão criados dois *WrapPanels* dentro de *Canvas*, no primeiro *WrapPanel* será utilizado para colocar os botões com os números e o segundo para os botões dos operadores (Figura 57).

```
<Canvas Margin="0,0,-8,-21">

    <TextBlock TextAlignment="Right" FontSize="30" x:Name="visor" Width="290" Height="50" Background="White" Canvas.Left="8" Canvas.Top="10"></TextBlock>

    <WrapPanel HorizontalAlignment="Left" Height="157" VerticalAlignment="Top" Width="171" Canvas.Left="19" Canvas.Top="68" >
        <Button x:Name="bt1" Content="1" Width="46" Height="34" Margin="2" Click="btnum_Click"/>
        <Button x:Name="bt2" Content="2" Width="46" Height="34" Margin="2" Click="btnum_Click"/>
        <Button x:Name="bt3" Content="3" Width="46" Height="34" Margin="2" Click="btnum_Click"/>
        <Button x:Name="bt4" Content="4" Width="46" Height="34" Margin="2" Click="btnum_Click"/>
        <Button x:Name="bt5" Content="5" Width="46" Height="34" Margin="2" Click="btnum_Click"/>
        <Button x:Name="bt6" Content="6" Width="46" Height="34" Margin="2" Click="btnum_Click"/>
        <Button x:Name="bt7" Content="7" Width="46" Height="34" Margin="2" Click="btnum_Click"/>
        <Button x:Name="bt8" Content="8" Width="46" Height="34" Margin="2" Click="btnum_Click"/>
        <Button x:Name="bt9" Content="9" Width="46" Height="34" Margin="2" Click="btnum_Click"/>
        <Button x:Name="bt0" Content="0" Width="46" Height="34" Margin="2" Click="btnum_Click"/>
        <Button x:Name="btponto" Content="," Width="46" Height="34" Margin="2" Click="btnum_Click"/>
        <Button x:Name="btnega" Content="-" Width="46" Height="34" Margin="2" Click="btnega_Click"/>
    </WrapPanel>

    <WrapPanel Height="157" Canvas.Left="177" Canvas.Top="68" Width="100">
        <Button x:Name="btmult" Content="X" Width="46" Height="34" Margin="2" Click="btopera_Click" />
        <Button x:Name="btsoma" Content="+" Width="46" Height="34" Margin="2" Click="btopera_Click" />
        <Button x:Name="btdivid" Content="/" Width="46" Height="34" Margin="2" Click="btopera_Click"/>
        <Button x:Name="btsub" Content="-" Width="46" Height="34" Margin="2" Click="btopera_Click"/>
        <Button x:Name="btC" Content="C" Width="46" Height="34" Margin="2" Click="btCLEAN_Click"/>
        <Button x:Name="btEuroMilhoes" Content="€uroM" Width="46" Height="34" Margin="2"/>
        <Button x:Name="btigual" Content="=" Width="97" Height="34" Margin="2" Click="btigual_Click"/>
    </WrapPanel>
```

Figura 57: Construção dos Menus (Calculadora).

Fonte: do autor.

No *code behind* da *calculadora.xml* é necessário definir os métodos que serão tratados, criando um *delegate. D*entro da classe Calculadora sao criadas várias variáveis *bool, float* para limpar o visor e para os argumentos, tendo sempre em atenção que os mesmos poderão receber valores *Null*. Ao contrário dos eventos que têm sidos associados aos botões anteriores neste caso apenas é associado um único evento para os botões que se encontram no primeiro *WrapPanel* e no evento *btnum_Click* será lido o conteúdo do botão e lançado para o visor (Figura 58).

```
namespace WpfApp1
{
    public delegate float? dlgopera(float? a, float? b);
    public partial class Calculadora : Window
    {

        bool limpaVisor = true;
        bool primeiroArg = true;
        float? arg1, arg2;
        dlgopera func = null;

        public Calculadora()
        {
            InitializeComponent();
        }
        private void btnum_Click(object sender, RoutedEventArgs e)
        {
            Button bt = (Button)sender;
            if (limpaVisor) { visor.Text = ""; limpaVisor = false; }
            visor.Text += bt.Content.ToString();
        }
```

Figura 58: Evento dos botões.

Fonte: do autor.

No evento dos botões das operações *btopera_Click* é necessário ler o primeiro argumento, o utilizador insere o primeiro número e assim que carrega no operador ele

percorre um *switch* para efetuar a conta com o segundo argumento, caso o utilizador tente utilizar um operador sem ter argumentos será mostrado uma mensagem no visor a partir do *catch (Exception erro)* (Figura 59).

```
private void btopera_Click(object sender, RoutedEventArgs e)
        {
            Button bt = (Button)sender;
            try
            {if(primeiroArg)
                {
                    arg1 = float.Parse(visor.Text);
                    primeiroArg = false;
                    limpaVisor = true;
                }else
                {
                if(func !=null && !string.IsNullOrEmpty(visor.Text))
                    {
                        arg2 = float.Parse(visor.Text);
                        arg1 = func(arg1, arg2);
                        visor.Text = arg1.ToString();
                        arg2 = null;
                        limpaVisor = true;
                    }}
                switch(bt.Content.ToString())
                {
                    case "+":
                        func = (a, b) => a + b;
                        break;
                    case "-":
                        func = (a, b) => a - b;
                        break;
                    case "/":
                        func = (a, b) => a / b;
                        break;
                    case "*":
                        func = (a, b) => a * b;
                        break;
                }
            }
            catch (Exception erro)
            {

                visor.Text = erro.Message;
            }
        }
```

Figura 59: Função para as operações.

Fonte: do autor.

No evento do botão "C" (*clean*), será necessário configurar de modo a limpar o visor e os argumentos guardados anteriormente. Após configuração do botão "C" dá-se início à configuração do evento para o botão de igual, para tal é necessário associar o valor da variável *func* que guarda a operação feita pelas duas variáveis e mostra-as no visor (Figura 60).

```
private void btCLEAN_Click(object sender, RoutedEventArgs e)
        {
            visor.Text = "";
            arg1 = null;
            arg2 = null;
            limpaVisor = true;
            primeiroArg = true;
            func = null;
        }
        private void btigual_Click(object sender, RoutedEventArgs e)
        {
            try
            {
             if(arg1 !=null && !string.IsNullOrEmpty(visor.Text)&& func !=null)
                {
                    arg2 = float.Parse(visor.Text);
                    visor.Text = func(arg1, arg2).ToString();
                    arg2 = arg2 = null;
                    func = null;
                    limpaVisor = true;
                    primeiroArg = true;
                }
            }
            catch (Exception erro)
            {
                visor.Text = erro.Message;
            }
        }
```

Figura 60: Botão de Clean e igual.

Fonte: do autor.

Para a calculadora poder trabalhar com números negativos utilizou-se o símbolo de menos que ficou situado no primeiro *WrapPanel* para não confundir com a subtração. Para se utilizar números negativos é necessário ter o argumento no visor para funcionar, caso contrário aparece erro no ecrã (Figura 61).

```
        private void btnega_Click(object sender, RoutedEventArgs e)
        {
            try
            {
                float? rslt = float.Parse(visor.Text);
                rslt *= -1;
                visor.Text = rslt.ToString();
            }
            catch (Exception erro)
            {
                visor.Text = erro.Message;
            }
        }
```

Figura 61: Números negativos.

Fonte: do autor.

Janela de Pagamento

A janela de pagamento será o último separador do programa desenvolvido e, por sua vez, o mais complexo. O separador irá recolher os dados do cliente a partir de 7 *TexBox*'s (*ID*, Nome, Morada, Localidade, CC, NIF, e a Foto) com as respetivas *Labels*. Posteriormente, é adicionada uma *Textbox* para receber o caminho da foto.

Depois de introduzidas as caixas de texto, estas vão receber os dados do cliente, que por sua vez são adicionados a um *DataGrid* para receber os valores que vêm da base de dados. São adicionados, também, 5 botões: o primeiro, "Salvar Dados" que permite guardar os dados inseridos e guardá-los na base de dados; seguido do "Actualizar" para fazer o *update* aos dados já inseridos na base de dados; o botão "Procurar" permitirá que o utilizador insira uma foto; e, por último, o botão de "*PDF*" que vai permitir exportar os dados para o formato (*.pdf)* (Figura 62 e 63).

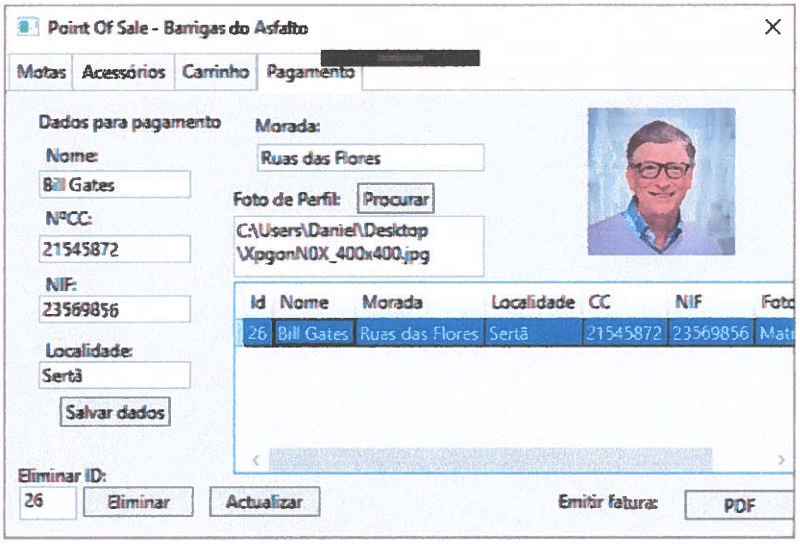


Figura 62: Separador de Pagamentos.

Fonte: do autor.

Figura 63: Separador de Pagamentos (XML).

Fonte: do autor.

Depois de construída a parte gráfica do separador pagamentos, inicia-se a configuração dos botões e a sua integração com a base de dados. No evento do botão "Salvar Dados" é necessário a partir da ligação à base de dados passar todos os valores das caixas de texto para a base de dados. No entanto, a caixa de texto que irá receber imagem terá um tratamento especial, pois caso não seja escolhida nenhuma imagem será guardado um valor *null* porém, caso o utilizador escolha uma imagem a mesma tem de ser tratada a partir de uma função que converta para bitmaps e guarde na base de dados (Figura 64 e 65).

```
private void Salvardados_Click(object sender, RoutedEventArgs e)
{
    using (Loja1Entities6 db = new Loja1Entities6())
    {
        byte[] image = null;
        if (!string.IsNullOrEmpty(txtcaminho.Text))
        {
            image = utils.ConvertBitmapSourceToByteArray(txtcaminho.Text);
        }
        Cliente1 mycliente = new Cliente1
        {
            Nome = Tbnome.Text,
            Morada = tbmorada.Text,
            Localidade = tblocalidade.Text,
            NIF = tbnif.Text,
            CC = tbcc.Text,
            Foto = image
        };
        db.Cliente1.Add(mycliente);
        db.SaveChanges();
        gridpagamento.ItemsSource = db.Cliente1.ToList();
    }

}
```

Figura 64:Evento do Botão Salvar dados

Fonte: do autor.

```
public static byte[] ConvertBitmapSourceToByteArray(string filepath)
{
    var image = new BitmapImage(new Uri(filepath));
    byte[] data;
    BitmapEncoder encoder = new JpegBitmapEncoder();
    encoder.Frames.Add(BitmapFrame.Create(image));
    using (MemoryStream ms = new MemoryStream())
    {
        encoder.Save(ms);
        data = ms.ToArray();
    }
    return data;
}
```

Figura 65: Função Convert to Bitmap.

Fonte: do autor.

No evento do botão de "Eliminar" o processo é semelhante ao anterior, contudo será necessário ler o valor introduzido na caixa de texto (*tbdelete*), depois é criada uma *string* ID que irá receber o valor e será comparada com a ID que existe na base de dados,

caso exista, será efetuado o processo de "*remove*", isto é, elimina o registo da base de dados (Figura 66).

```
  private void btdelete_Click(object sender, RoutedEventArgs e)
        {
            using (Loja1Entities6 db = new Loja1Entities6())
            {
                int idmota;
                if (int.TryParse(tbidmota.Text, out idmota))
                {
                    tbmota mymota = db.tbmotas.Where(x => x.idmota ==
idmota).FirstOrDefault();
                    if (mymota != null) db.tbmotas.Remove(mymota);
                    else MessageBox.Show("Este registo já foi eliminado!");
                    db.SaveChanges();
                    double? valorTotal = db.tbmotas.Sum(s => s.custo);


                    totalPrice.Text = valorTotal.ToString();
                    grdListaTudo.ItemsSource = db.tbmotas.ToList();
                }
            }
```

Figura 66: Evento do botão Eliminar.

Fonte: do autor.

```
private void insert_click(object sender, RoutedEventArgs e)
        {

            using (Loja1Entities6 db = new Loja1Entities6())
            {
                int.TryParse(tbdelete.Text, out int a);
                var mycliente = db.Cliente1.Where(w => w.id == a).FirstOrDefault();
                mycliente.Nome = Tbnome.Text.ToString();
                mycliente.Morada = tbmorada.Text.ToString();
                mycliente.Localidade = tblocalidade.Text.ToString();
                mycliente.NIF = tbnif.Text.ToString();
                mycliente.CC = tbcc.Text.ToString();
                if (!string.IsNullOrEmpty(txtcaminho.Text))
                {
                    mycliente.Foto = utils.ConvertBitmapSourceToByteArray(txtcaminho.Text);
                }

                db.SaveChanges();
                gridpagamento.ItemsSource = db.Cliente1.ToList();

            }
        }
```

Figura 67: Evento do Botão Actualizar.

Fonte: do autor.

O evento do botão actualizar é semelhante ao *delete*, através da caixa de texto tbdelete é colocado o *ID* do utilizador que se pretende editar e é feita uma comparação com o id existente na base de dados e os valores das restantes caixas de texto modificam os dados já existentes (Figura 67).

É adicionado também na *grid* o evento (*PreviewMouseDoubleClick*), que permite que assim que se pressione duas vezes com o rato em cima da linha que se pretende apenas mostre os dados inseridos e a imagem em questão. Assim que é pressionado na linha os valores da linha são passados para as caixas de texto permitindo a sua edição novamente sendo sempre necessário o botão de atualizar para guardar na base de dados. A *grid* é

colocada na opção (*readonly = true*) para que quando seja carregado em cima não os edite os campos, mas sim os mostre (Figura 68).

```
    private void gridpagamento_PreviewMouseDoubleClick(object sender, MouseButtonEventArgs e)
        {
            gridpagamento.IsReadOnly = true;
            DataGrid d = (DataGrid)sender;
            int indexRow = d.SelectedIndex;
 var id =
((WpfApp1.Cliente1)((System.Windows.Controls.ItemsControl)sender).Items[indexRow]).id;
            using (Loja1Entities6 db = new Loja1Entities6())
            {
                var cliente = db.Cliente1.Where(w => w.id == id).FirstOrDefault();
                tbdelete.Text = cliente.id.ToString();
                Tbnome.Text = cliente.Nome;
                tbmorada.Text = cliente.Morada;
                tblocalidade.Text = cliente.Localidade;
                tbnif.Text = cliente.NIF;
                tbcc.Text = cliente.CC;
                if (cliente.Foto != null)
                {
                    imagem.Source = utils.ConvertByteArrayToBitmapImage(cliente.Foto);
                } } }
```

Figura 68: Evento da Grid pagamentos.

Fonte: do autor.

Por ultimo o evento do botão PDF, o seu objetivo é utilizar os dados inseridos nas tabelas (tbmotas e Clientes1) e criar um recibo do tipo pdf. Para poder exportar os dados para PDF foi utilizada uma biblioteca da *iTextsharp* instalada a partir da loja do Windows dentro do *Visual Studio*, esta biblioteca contem um conjunto de ferramentas que permite criar e elaborar um pdf. Após instalado o iTextSharp é necessário utilizar a seguinte bibliotecas no topo da *MainWindow* (*using iTextSharp.text; using iTextSharp.text.pdf; using System.Diagnostics*)(Figura 69 e 70).

```
private void PDF_Click(object sender, RoutedEventArgs e)
        {Document doc = new Document(PageSize.A4);
            string dir = @"C:\PDF" + @"\pagamento.pdf";
            var output = new FileStream(dir, FileMode.Create);
            var writer = PdfWriter.GetInstance(doc, output);
            doc.Open();
            PdfPTable table = new PdfPTable(7);
            table.DefaultCell.BorderColor = BaseColor.BLACK;
            table.DefaultCell.HorizontalAlignment = Element.ALIGN_CENTER;
            PdfPCell cell = new PdfPCell();
            cell.Colspan = 7;
            cell.HorizontalAlignment = Element.ALIGN_CENTER;
            cell.VerticalAlignment = Element.ALIGN_CENTER;
            table.AddCell(cell);
            table.AddCell(new Phrase("Id"));
            table.AddCell(new Phrase("Nome"));
            table.AddCell(new Phrase("Morada"));
            table.AddCell(new Phrase("Localidade"));
            table.AddCell(new Phrase("CC"));
            table.AddCell(new Phrase("NIF"));
            table.AddCell(new Phrase("Foto"));
            using (Loja1Entities6 db = new Loja1Entities6())
            {foreach (Cliente1 i in db.Cliente1)
                {
                    table.AddCell(new Phrase(i.id.ToString()));
                    table.AddCell(new Phrase(i.Nome.ToString()));
                    table.AddCell(new Phrase(i.Morada.ToString()));
                    table.AddCell(new Phrase(i.Localidade.ToString()));
                    table.AddCell(new Phrase(i.NIF.ToString()));
                    table.AddCell(new Phrase(i.CC.ToString()));
                    var foto = iTextSharp.text.Image.GetInstance(i.Foto);
                    foto.ScaleAbsoluteHeight(200);
                    foto.ScaleAbsoluteWidth(170);
                    table.AddCell(foto);
                }
            }
```

Figura 69: Evento do botão de PDF.

Fonte: do autor.

```
var foto = iTextSharp.text.Image.GetInstance(i.Foto);
                    foto.ScaleAbsoluteHeight(200);
                    foto.ScaleAbsoluteWidth(170);
                    table.AddCell(foto);
                }
            }
            doc.Add(table);
            doc.Add(new Phrase("\n\n"));
            PdfPTable table1 = new PdfPTable(5);
            table1.DefaultCell.BorderColor = BaseColor.BLACK;
            table1.DefaultCell.HorizontalAlignment = Element.ALIGN_CENTER;
            PdfPCell cell1 = new PdfPCell();
            cell1.Colspan = 5;
            cell1.HorizontalAlignment = Element.ALIGN_CENTER;
            cell1.VerticalAlignment = Element.ALIGN_CENTER;
            table1.AddCell(cell1);
            table1.AddCell(new Phrase("idmota"));
            table1.AddCell(new Phrase("Mota"));
            table1.AddCell(new Phrase("Acessorios"));
            table1.AddCell(new Phrase("Valor"));
            table1.AddCell(new Phrase("c/Iva"));
            using (Loja1Entities6 db = new Loja1Entities6())
            {
                foreach (tbmota i in db.tbmotas)
                {
                    table1.AddCell(new Phrase(i.idmota.ToString()));
                    table1.AddCell(new Phrase(i.mota.ToString()));
                    table1.AddCell(new Phrase(i.acessorios.ToString()));
                    table1.AddCell(new Phrase(i.custo.ToString()));
                    table1.AddCell(new Phrase(i.custoiva.ToString()));
                }
            }
            doc.Add(table1);
            doc.Close();
            Process.Start(dir);
        }
```

Figura 70: Evento do botão de PDF.

Fonte: do autor.

Por fim na Figura 71 é demonstrado o recibo em pdf gerado pela aplicação, foram introduzidos alguns dados na tabela para demonstração, o mesmo será guardo numa pasta destinada ao efeito (C:/PDF).

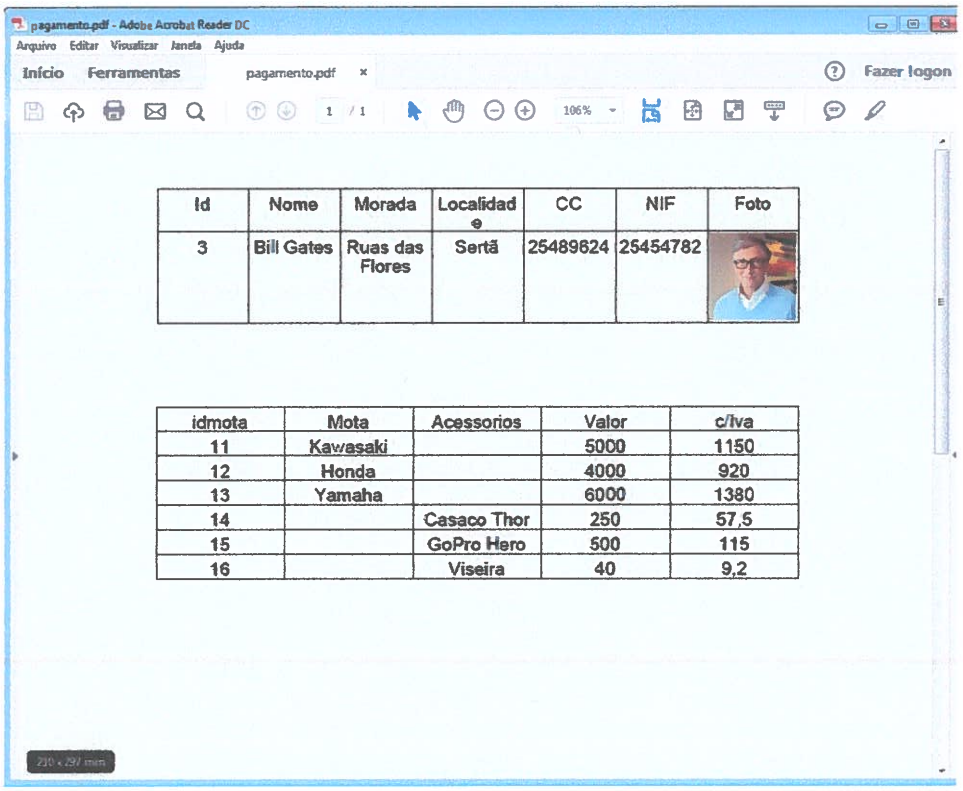


Figura 71: Exportação para PDF.

Fonte: do autor.

Conclusão

Com o projeto terminado consegue-se agora ter uma melhor perceção da importância que a virtualização tem nos dias de hoje e como podemos gerir os recursos dos equipamentos tirando assim um melhor rendimento. A realização deste projeto permitiu fortalecer algumas das matérias lecionadas ao longo da licenciatura e à consolidação dos conhecimentos. Os resultados alcançados foram notáveis pois, uma vez que com um equipamento de "consumo" produziu-se um ambiente empresarial totalmente viável. A construção do *domain controller* com o serviço *Network Policie Service* tornou a rede mais segura, em caso de incumprimento dos requisitos da máquina cliente, fica sem qualquer contacto à rede, isolando por completo qualquer tipo de ameaça até à resolução das anomalias. A configuração do servidor de *Routing* como único ponto de acesso ao exterior torna ainda mais segura a rede interna, protegendo assim contra-ataques. A sua fácil configuração torna-se um ponto vital para as empresas que querem assegurar as suas comunicações. A elaboração da parte de desenvolvimento é também um dos resultados alcançados mais importante pois não sendo uma das minhas áreas de conforto, levou-me a um estudo mais aprofundado relativamente à *framework* ADO.net.

Algumas das dificuldades sentidas no desenvolvimento deste projeto foi a incompatibilidade com os controladores do *host* em questão, visto ser um processador AMD, e tecnologia *Radeon*. A gestão dos recursos foi também uma das dificuldades sentidas, visto que o *host* em questão foi adquirido em 2008, devido a alguns problemas a nível de desempenho.

A aplicação desenvolvida permite alterações e novas funcionalidades, um dos pontos a ser melhorado é ao nível de avisos para o cliente. Foram implementados alguns alertas em caso de eliminar o mesmo registo, entre outros, contudo existem outros campos em que não existe alerta. Uma das melhorias a implementar na aplicação seria o envio do recibo de pagamento através de email. A divisão da aplicação entre utilizador corrente e administrador seria uma mais valia, pois o cliente teria acesso a um tipo de dados e o administrador teria outros privilégios.

Acrónimos

i. ADO - Activex Data Objects
ii. AMD - Advanced Micro Devices
iii. BIOS - Basic Input Output System
iv. C# - C-Sharp
v. DC - Domain Controller
vi. DHCP - Dynamic Host Configuration Protocol
vii. DNS – Domain Name System
viii. FQDN - Fully Qualified Domain Name
ix. IP - Internet Protocol
x. NAP - Network Access Protection
xi. NPS - Network Policy Server
xii. OU - Organizational Unit
xiii. SO - Sistema Operativo
xiv. SQL - Structured Query Language
xv. VHDX - Virtual Hard Disk/X
xvi. VM – Virtual Machine
xvii. WPF - Windows Presentation Foundation
xviii. XAML - Extensible Application Markup Language
xix. XML - eXtensible Markup Language

Referências Bibliográficas

Calbimonte, D. (2018). *The history of SQL Server – The evolution of SQL Server features*. Retirado de https://www.sqlshack.com/history-sql-server-evolution-sql-server-features/ (Consultado dia 20 de Abril de 2018).

Carissima, A. (2009). *Virtualização: Princípios Básicos e Aplicações*. Retirado de http://www.lbd.dcc.ufmg.br/colecoes/erad/2009/004.pdf (Consultado dia 27 de Abril de 2018).

Desai, M. (2012). *High Performance Computing and Virtualization*. CSCi-555 *Advanced Operating Systems.* Paper do departamento da ciência dos computadores da universidade do Sul da Califórnia. Los Angeles.

Ferreira, N. (2004). *ADO.NET.* Departamento de Engenharia Informática no Instituto Superior de Engenharia do Porto. Porto. Retirado de http://www.dei.isep.ipp.pt/~jtavares/ADAV/downloads/guiao_ADO_dotNet.pdf

Microsoft (2015). *Server Virtualization Windows server 2012*. Proceedings da Microsoft.

Microsoft (2017a). *Microsoft SQL Server (2016), What's New in SQL Server 2016*. Retirado de https://docs.microsoft.com/en-us/sql/sql-server/what-s-new-in-sql-server-2016

Microsoft (2017b). *Microsoft SQL Server (2017) What's New in SQL Server 2017*. Retirado de https://docs.microsoft.com/en-us/sql/sql-server/what-s-new-in-sql-server-2017

Microsoft (2017c). *Microsoft SQL Server Maximum Capacity Specifications for SQL Serve,* Retirado de https://docs.microsoft.com/en-us/sql/sql-server/maximum-capacity-specifications-for-sql-server

Panek W. (2015). *MSCA Windows Server 2012 R2 COMPLETE STUDY GUIDE. SYBEX. ISBN: 978-1-118-85991-9*

Stanek, W. R. (2016). *Windows Server 2016 Essentials for Administration*, IT Pro Solutions. *ISBN*: 9781534601369

Microsoft (2016). Novo Painel de Controlo Windows Server 2016. [image] Available at: https://www.microsoft.com/pt-pt [Consultado em 2 Abril 2018].

Microsoft (2016). UAC - User Account Control Settings. [image] Available at: https://www.microsoft.com/pt-pt [Consultado em 9 Abril 2018].

Infordisa (2016). Diferença entre versões do Windows Server. [imagem] Disponível em: http://www.infordisa.com/es/ediciones-de-windows-server-2016/ [Consultado em 8 Abril 2018].

Amavsharma (2014). Comparação entre hyper-V 2008 e 2012. [imagem] Disponível em: https://arnavsharma.net/windows-server/hyper-v-windows-server-2012-r2-vs-2008-r2 [Consultado em 11 Abril 2018].

Microsoft (2017). Virtual Network Manager. [imagem] Disponível em: https://docs.microsoft.com/en-us/virtualization/hyper-v-on-windows/quick-start/connect-to-network [Consultado em 1 Maio 2018].

Hackerworld (2016). A evolução do ASP.NET. [imagem] Disponível em: https://hackerworld.co/asp-net-core-1-0-introduction-general-description-and-future-of-net-framework [Consultado em 10 Maio 2018].

Serra, J. (2016). Evolução dos SQL Server. [imagem] Disponível em: http://www.sqlmvp.org/microsoft-sql-server-evolution-from-2000-to-2016/ [Consultado em 18 Maio 2018].

Microsoft (2017). Limites Máximos do SQL Server 2017. [imagem] Disponível em: https://docs.microsoft.com/en-us/sql/sql-server/maximum-capacity-specifications-for-sql-server?view=sql-server-2017 [Consultado em 10 Maio 2018].

Anexos

Anexo I

XAML da Janela de Autenticação

```
<Window x:Class="WpfApp1.MainWindow"
        xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
        xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
        xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
        xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
        xmlns:local="clr-namespace:WpfApp1"
        mc:Ignorable="d"
        Title="POS-Barrigas Do Asfalto" Height="278" Width="353"
ResizeMode="NoResize" Icon="C:\Barrigas 2\wpfapp1\WpfApp1\Imagens\ico.png">
    <Grid Margin="-74,0,-70,-270">
        <TextBox x:Name="txtuser"  HorizontalAlignment="Left" Height="23"
Margin="202,108,0,0" TextWrapping="Wrap" Text="" VerticalAlignment="Top"
Width="155"/>
        <PasswordBox x:Name="txtpassword" HorizontalAlignment="Left"
Margin="202,162,0,0" VerticalAlignment="Top" Width="155" Height="22"/>
        <Label Content="Utilizador:" HorizontalAlignment="Left"
Margin="200,77,0,0" VerticalAlignment="Top" RenderTransformOrigin="0.508,1.071"/>
        <Label Content="Password:" HorizontalAlignment="Left"
Margin="201,136,0,0" VerticalAlignment="Top"
RenderTransformOrigin="1.319,0.351"/>
        <Button x:Name="Btlogin" Content="Login" HorizontalAlignment="Left"
Margin="202,189,0,0" VerticalAlignment="Top" Width="75" Click="Btlogin_Click"
RenderTransformOrigin="0.467,-0.15"/>
        <Button Content="Sair" HorizontalAlignment="Left" Margin="282,189,0,0"
VerticalAlignment="Top" Width="75" Click="Button_Click"/>

        <Image HorizontalAlignment="Left" Height="100" Margin="96,93,0,0"
VerticalAlignment="Top" Width="100" Source=".\Imagens\lock.png"/>
        <Image HorizontalAlignment="Left" Height="78" Margin="113,10,0,0"
VerticalAlignment="Top" Width="274" RenderTransformOrigin="0.36,0.25"
Source=".\Imagens\logo4.jpg"/>
    </Grid>
</Window>
```

Anexo II

Código Fonte da Janela de Autenticação

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows;
using System.Windows.Forms;
using System.Windows.Controls;
using System.Windows.Data;
using System.Windows.Documents;
using System.Windows.Input;
using System.Windows.Media;
using System.Windows.Media.Imaging;
using System.Windows.Navigation;
using System.Windows.Shapes;
using System.Data.SqlClient;
using System.Data.Sql;
```

```
using System.Data;
using System.Runtime.InteropServices;

namespace WpfApp1
{
    /// <summary>
    /// Interação lógica para MainWindow.xam
    /// </summary>
    public partial class MainWindow : Window
    {
        public MainWindow()
        {
            InitializeComponent();
        }

        [DllImport("advapi32.dll")]
        public static extern bool LogonUser(string name, string domain, string
pass, int logType, int logpv, ref IntPtr pht);
        private void Btlogin_Click(object sender, RoutedEventArgs e)
        {
            IntPtr th = IntPtr.Zero;
            bool log = LogonUser(txtuser.Text, "pg.local", txtpassword.Password,
2, 0, ref th);
            if (log)
            {
                this.Hide();
                Menus ss = new Menus();
                ss.Show();
            }

        }
        private void Button_Click(object sender, RoutedEventArgs e)
        {
            this.Close();
        }

    }
}
```

Anexo III

XAML da Janela de Menus

```
<Window x:Class="WpfApp1.Menus"

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

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

      xmlns:d="http://schemas.microsoft.com/expression/blend/2008"

      xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"

      xmlns:local="clr-namespace:WpfApp1"

      mc:Ignorable="d"
```

```
    Title="Point Of Sale - Barrigas do Asfalto" Height="371" Width="554"
ResizeMode="NoResize" Icon="C:\Barrigas 2\wpfapp1\WpfApp1\Imagens\ico.png">
  <Grid>
    <TabControl>
      <TabItem Header="Motas">
        <Grid Background="#FFE5E5E5" Margin="0,-2,-3,-2"
RenderTransformOrigin="0.5,0.5">
          <Grid.RenderTransform>
            <TransformGroup>
              <ScaleTransform/>
              <SkewTransform/>
              <RotateTransform Angle="0.046"/>
              <TranslateTransform/>
            </TransformGroup>
          </Grid.RenderTransform>
          <Button x:Name="Honda" HorizontalAlignment="Left"
Margin="29,151,0,0" VerticalAlignment="Top" Width="75" Height="83"
Click="Honda_Click">
            <Button.Template>
              <ControlTemplate>
                <Image Source="/imagens/cbr.jpg"/>
              </ControlTemplate>
            </Button.Template>

          </Button>
```

```
            <Button x:Name="KTM" HorizontalAlignment="Left"
Margin="29,34,0,0" VerticalAlignment="Top" Width="75" Height="81"
Click="KTM_Click">
                <Button.Template>
                  <ControlTemplate>
                    <Image Source="/imagens/ktm.jpg"/>
                  </ControlTemplate>
                </Button.Template>
            </Button>
            <Button x:Name="Kawasaki" Content="Button"
HorizontalAlignment="Left" Margin="153,34,0,0" VerticalAlignment="Top"
Width="75" Height="81" Click="Kawasaki_Click">
                <Button.Template>
                  <ControlTemplate>
                    <Image Source="/imagens/kawasaki.jpg"/>
                  </ControlTemplate>
                </Button.Template>
            </Button>
            <Button x:Name="BMW" Content="Button" HorizontalAlignment="Left"
Margin="153,151,0,0" VerticalAlignment="Top" Width="75" Height="83"
Click="BMW_Click">
                <Button.Template>
                  <ControlTemplate>
                    <Image Source="/imagens/bmw.jpg"/>
                  </ControlTemplate>
                </Button.Template>
```

```
            </Button>
            <Button x:Name="Suzuki" Content="Button" HorizontalAlignment="Left"
Margin="279,151,0,0" VerticalAlignment="Top" Width="75" Height="83"
Click="Suzuki_Click">
                <Button.Template>
                    <ControlTemplate>
                        <Image Source="/imagens/suzuki.jpg"/>
                    </ControlTemplate>
                </Button.Template>
            </Button>
            <Label Content="KTM Duke " HorizontalAlignment="Left"
Margin="17,120,0,0" VerticalAlignment="Top" Width="119"
RenderTransformOrigin="0.723,0.346"/>
            <Label Content="4000€" HorizontalAlignment="Left"
Margin="74,120,0,0" VerticalAlignment="Top" Width="119"/>
            <Label Content="KAWASAKI 5000€&#xD;&#xA;"
HorizontalAlignment="Left" Margin="153,120,0,0" VerticalAlignment="Top"/>
            <Label Content="YAMAHA 6000€" HorizontalAlignment="Left"
Margin="279,120,0,0" VerticalAlignment="Top"/>
            <Label Content="HONDA 5500€&#xD;&#xA;"
HorizontalAlignment="Left" Margin="29,239,0,0" VerticalAlignment="Top"
Height="26"/>
            <Label Content="BMW 14000€" HorizontalAlignment="Left"
Margin="153,239,0,0" VerticalAlignment="Top"/>
            <Label Content="SUZUKI 2000€" HorizontalAlignment="Left"
Margin="279,239,0,0" VerticalAlignment="Top"/>
```

```
            <Button Content="Finalizar" HorizontalAlignment="Left"
Margin="300,271,0,0" VerticalAlignment="Top" Width="75" Click="Button_Click_1"
/>

            <Button Content="Acessórios" HorizontalAlignment="Left"
Margin="380,271,0,0" VerticalAlignment="Top" Width="75"
Click="Button_Click_2"/>

            <Label Content="Motas Disponiveis&#xD;&#xA;"
HorizontalAlignment="Left" Margin="29,3,0,0" VerticalAlignment="Top"
Height="26"/>

            <Button x:Name="Yamaha" HorizontalAlignment="Left"
Margin="279,34,0,0" VerticalAlignment="Top" Width="75" Height="86"
Click="Yamaha_Click">

              <Button.Template>

                <ControlTemplate>

                  <Image Source="/imagens/mt.jpg"/>

                </ControlTemplate>

              </Button.Template>

            </Button>

            <Button Content="Logout" HorizontalAlignment="Left"
Margin="460,271,0,0" VerticalAlignment="Top" Width="75" Click="Button_Click"/>

            <DataGrid Name="gridmotas"  HorizontalAlignment="Left" Height="239"
Margin="372,10,0,0" VerticalAlignment="Top" Width="161" >

              <DataGrid.Columns>

                <DataGridTextColumn Header="Mota" Binding="{Binding mota}" />

                <DataGridTextColumn Header="Acessorios" Binding="{Binding
acessorios}" />

                <DataGridTextColumn Header="c/Iva" Binding="{Binding
custoiva}"/>
```

```
            </DataGrid.Columns>
          </DataGrid>
        </Grid>
      </TabItem>
      <TabItem Header="Acessórios" x:Name="tabacessorios" Height="24"
VerticalAlignment="Bottom" Margin="-2,0,-2,1" >
        <Grid Background="#FFE5E5E5" Margin="0,-2,0,-3">
          <Grid.ColumnDefinitions>
            <ColumnDefinition Width="6"/>
            <ColumnDefinition Width="8*"/>
            <ColumnDefinition Width="259*"/>
          </Grid.ColumnDefinitions>
          <Button Content="Button" HorizontalAlignment="Left"
Margin="7,151,0,0" VerticalAlignment="Top" Width="75" Height="83"
Grid.Column="2" Click="Button_luvas" >
            <Button.Template>
              <ControlTemplate>
                <Image Source="/imagens/luvasjpg.jpg"/>
              </ControlTemplate>
            </Button.Template>
          </Button>
          <Button Content="Button" HorizontalAlignment="Left"
Margin="7,34,0,0" VerticalAlignment="Top" Width="75" Height="81"
Grid.Column="2" Click="Button_Capacete">
            <Button.Template>
              <ControlTemplate>
```

```
                    <Image Source="/imagens/capacete.jpg"/>
                </ControlTemplate>
            </Button.Template>
        </Button>
        <Button Content="Button" HorizontalAlignment="Left"
Margin="257,34,0,0" VerticalAlignment="Top" Width="75" Height="81"
Grid.Column="2" Click="Button_botas">
            <Button.Template>
                <ControlTemplate>
                    <Image Source="/imagens/botas.jpg"/>
                </ControlTemplate>
            </Button.Template>
        </Button>
        <Button Content="Button" HorizontalAlignment="Left"
Margin="131,34,0,0" VerticalAlignment="Top" Width="76" Height="81"
Grid.Column="2" Click="Button_Casaco">
            <Button.Template>
                <ControlTemplate>
                    <Image Source="/imagens/casaco.jpg"/>
                </ControlTemplate>
            </Button.Template>
        </Button>
        <Button Content="Button" HorizontalAlignment="Left"
Margin="131,151,0,0" VerticalAlignment="Top" Width="75" Height="83"
Grid.Column="2" Click="Button_Camara">
            <Button.Template>
```

```
                <ControlTemplate>
                    <Image Source="/imagens/gopro.jpg"/>
                </ControlTemplate>
            </Button.Template>
        </Button>
        <Button Content="Button" HorizontalAlignment="Left"
Margin="257,151,0,0" VerticalAlignment="Top" Width="75" Height="83"
Grid.Column="2" Click="Button_viseira">
            <Button.Template>
                <ControlTemplate>
                    <Image Source="/imagens/viseira.jpg"/>
                </ControlTemplate>
            </Button.Template>
        </Button>
        <Label Content="Shark 500€" HorizontalAlignment="Left"
Margin="7,120,0,0" VerticalAlignment="Top" Grid.Column="2" Height="26"
Width="69"/>
        <Label Content="Alpinestar 200€" HorizontalAlignment="Left"
Margin="131,120,0,0" VerticalAlignment="Top" Grid.Column="2" Height="26"
Width="92"/>
        <Label Content="Botas 70€" HorizontalAlignment="Left"
Margin="257,120,0,0" VerticalAlignment="Top" Grid.Column="2" Height="26"
Width="62"/>
        <Label Content="Luvas 50€" HorizontalAlignment="Left"
Margin="7,239,0,0" VerticalAlignment="Top" Grid.Column="2" Height="26"
Width="62"/>
```

```
            <Label Content="GOPRO 400€" HorizontalAlignment="Left"
Margin="131,239,0,0" VerticalAlignment="Top" Grid.Column="2" Height="26"
Width="79"/>

            <Label Content="Viseira 20€" HorizontalAlignment="Left"
Margin="257,239,0,0" VerticalAlignment="Top" Grid.Column="2" Height="26"
Width="68"/>
            <Button Content="Finalizar" HorizontalAlignment="Left"
Margin="336,282,0,0" VerticalAlignment="Top" Width="75" Click="Button_Click_3"
Grid.Column="2" Height="20"/>

            <Label Content="Acessorios:&#xD;&#xA;" HorizontalAlignment="Left"
Margin="7,5,0,0" VerticalAlignment="Top" Grid.Column="2" Height="24"
Width="69"/>

            <Button Content="Logout" HorizontalAlignment="Left"
Margin="416,282,0,0" VerticalAlignment="Top" Width="75" Click="Button_logout"
Grid.Column="2" Height="20"/>

            <DataGrid Name="gridacessorios" Grid.Column="2"
HorizontalAlignment="Left" Height="239" Margin="347,11,0,0"
VerticalAlignment="Top" Width="161">

                <DataGrid.Columns>

                    <DataGridTextColumn Header="Mota" Binding="{Binding mota}" />

                    <DataGridTextColumn Header="Acessorios" Binding="{Binding
acessorios}" />

                    <DataGridTextColumn Header="c/Iva" Binding="{Binding
custoiva}"/>

                </DataGrid.Columns>

            </DataGrid>

        </Grid>

    </TabItem>

    <TabItem Header="Carrinho" x:Name="tabcarrinho">
```

```
        <Grid Background="#FFE5E5E5" Margin="0,-3,0,-2">
            <Grid.ColumnDefinitions>
                <ColumnDefinition Width="494*"/>
            </Grid.ColumnDefinitions>
            <Grid HorizontalAlignment="Left" Height="100" Margin="256,157,0,0"
VerticalAlignment="Top" Width="100"/>
            <DataGrid Name="grdListaTudo" CanUserAddRows="False"
AutoGenerateColumns="False" Margin="10,13,149,57"
SelectionChanged="grdListaTudo_SelectionChanged">
                <DataGrid.Columns>
                    <DataGridTextColumn Header="idmota" Binding="{Binding
idmota}" />
                    <DataGridTextColumn Header="Mota" Binding="{Binding mota}" />
                    <DataGridTextColumn Header="Acessorios" Binding="{Binding
acessorios}" />
                    <DataGridTextColumn Header="Valor" Binding="{Binding custo}"
/>
                    <DataGridTextColumn Header="c/Iva" Binding="{Binding
custoiva}"/>                </DataGrid.Columns>
            </DataGrid>
            <Label Content="Total:" HorizontalAlignment="Left"
Margin="306,279,0,0" VerticalAlignment="Top"/>
            <TextBlock HorizontalAlignment="Left" Margin="344,283,0,0"
VerticalAlignment="Top" Name="totalPrice" RenderTransformOrigin="0.5,0.5"
Width="78" Height="16">
                <TextBlock.RenderTransform>
                    <TransformGroup>
```

```
                    <ScaleTransform ScaleX="-1" ScaleY="-1"/>
                    <SkewTransform/>
                    <RotateTransform Angle="179.091"/>
                    <TranslateTransform/>
                  </TransformGroup>
                </TextBlock.RenderTransform>
              </TextBlock>
              <TextBox Name="tbmota" HorizontalAlignment="Left" Height="23"
Margin="398,76,0,0" TextWrapping="Wrap" VerticalAlignment="Top" Width="120"
/>              <TextBox Name="tbacessorio" HorizontalAlignment="Left"
Height="23" Margin="398,120,0,0" TextWrapping="Wrap" VerticalAlignment="Top"
Width="120"/>
              <TextBox Name="tbvalor" HorizontalAlignment="Left" Height="23"
Margin="398,164,0,0" TextWrapping="Wrap" VerticalAlignment="Top"
Width="120"/>
              <Button Name="btinsert" Content="Inserir" HorizontalAlignment="Left"
Margin="416,192,0,0" VerticalAlignment="Top" Width="75" Click="btinsert_Click"/>
              <Button Name="btupdate" Content="Actualizar"
HorizontalAlignment="Left" Margin="416,217,0,0" VerticalAlignment="Top"
Width="75" Click="btupdate_Click"/>
              <Button Name="btdelete" Content="Eliminar"
HorizontalAlignment="Left" Margin="416,242,0,0" VerticalAlignment="Top"
Width="75" RenderTransformOrigin="0.44,1.25" Click="btdelete_Click"/>
              <Button x:Name="calculadora" Content="Duvida nas Contas? "
HorizontalAlignment="Left" Margin="10,279,0,0" VerticalAlignment="Top"
Width="129" Click="calculadora_Click" />
              <TextBlock HorizontalAlignment="Left" Margin="442,55,0,0"
TextWrapping="Wrap" Text="Mota" VerticalAlignment="Top"
RenderTransformOrigin="1.5,0.563"/>
```

```
            <TextBlock HorizontalAlignment="Left" Margin="430,99,0,0"
TextWrapping="Wrap" Text="Acessórios" VerticalAlignment="Top"
RenderTransformOrigin="0.571,1.812"/>

            <TextBlock HorizontalAlignment="Left" Margin="428,143,0,0"
TextWrapping="Wrap" Text="Preço s/Iva" VerticalAlignment="Top"/>

            <TextBlock HorizontalAlignment="Left" Margin="442,13,0,0"
TextWrapping="Wrap" VerticalAlignment="Top"
RenderTransformOrigin="0.303,0.501"><Run Text="ID"/><Run
Text="Mota"/></TextBlock>

            <TextBox x:Name="tbidmota" HorizontalAlignment="Left" Height="23"
Margin="398,32,0,0" TextWrapping="Wrap" VerticalAlignment="Top" Width="120" /

            <Button Click="Pagar_click" Content="PAGAR"
HorizontalAlignment="Left" Margin="455,279,0,0" VerticalAlignment="Top"
Width="75"/>

        </Grid>

    </TabItem>

    <TabItem x:Name="tabpagamento" Header="Pagamento"
HorizontalAlignment="Right" Margin="0" RenderTransformOrigin="0.556,1"
Width="72" Height="25" VerticalAlignment="Bottom">

        <Grid Background="#FFE5E5E5" Margin="-4,0,-6,0">

            <Grid.ColumnDefinitions>

                <ColumnDefinition Width="23*"/>

                <ColumnDefinition Width="527*"/>

            </Grid.ColumnDefinitions>

            <TextBox Name="Tbnome" HorizontalAlignment="Left" Height="19"
Margin="1,52,0,0" TextWrapping="Wrap" VerticalAlignment="Top" Width="104"
TextChanged="TextBox_TextChanged_1" Grid.Column="1"/>
```

<TextBox Name="tbcc" HorizontalAlignment="Left" Height="19" Margin="1,96,0,0" TextWrapping="Wrap" VerticalAlignment="Top" Width="104" RenderTransformOrigin="0.481,1.211" TextChanged="tbcc_TextChanged" Grid.Column="1"/>

<TextBox Name="tbmorada" HorizontalAlignment="Left" Height="19" Margin="150,34,0,0" TextWrapping="Wrap" VerticalAlignment="Top" Width="155" Grid.Column="1"/>

<TextBox Name="tblocalidade" HorizontalAlignment="Left" Height="19" Margin="1,182,0,0" TextWrapping="Wrap" VerticalAlignment="Top" Width="104" Grid.Column="1"/>

<TextBox Name="tbnif" HorizontalAlignment="Left" Height="19" Margin="1,137,0,0" TextWrapping="Wrap" VerticalAlignment="Top" Width="104" TextChanged="TextBox_TextChanged" RenderTransformOrigin="0.433,0.105" Grid.Column="1"/>

<TextBlock HorizontalAlignment="Left" Margin="6,33,0,0" TextWrapping="Wrap" Text="Nome:" VerticalAlignment="Top" RenderTransformOrigin="1.618,0.313" Grid.Column="1"/>

<TextBlock HorizontalAlignment="Left" Margin="6,76,0,0" TextWrapping="Wrap" Text="NºCC:" VerticalAlignment="Top" Grid.Column="1"/>

<TextBlock HorizontalAlignment="Left" Margin="149,13,0,0" TextWrapping="Wrap" VerticalAlignment="Top" RenderTransformOrigin="0.614,-0.312" Grid.Column="1"><Run Text="Morada"/><Run Text=":"/></TextBlock>

<TextBlock HorizontalAlignment="Left" Margin="6,166,0,0" TextWrapping="Wrap" VerticalAlignment="Top" Grid.Column="1"><Run Text="Localidade"/><Run Text=":"/></TextBlock>

<TextBlock HorizontalAlignment="Left" Margin="6,121,0,0" TextWrapping="Wrap" Text="NIF:" VerticalAlignment="Top" Grid.Column="1"/>

<TextBlock HorizontalAlignment="Left" Margin="134,63,0,0" TextWrapping="Wrap" VerticalAlignment="Top" Height="20" Width="80" Grid.Column="1"><Run Text="Foto de Perfil:"/><LineBreak/><Run/></TextBlock>

```
            <Button Click="Procurar_Click" Content="Procurar"
HorizontalAlignment="Left" Margin="219,61,0,0" VerticalAlignment="Top"
Width="52" Grid.Column="1"/>

            <Label Content="" HorizontalAlignment="Left" Margin="134,124,0,0"
VerticalAlignment="Top" RenderTransformOrigin="0.105,0.346" Width="171"
Grid.Column="1"/>

            <Image  Name="imagem" HorizontalAlignment="Left" Height="113"
Margin="335,10,0,0" VerticalAlignment="Top" Width="160" Grid.Column="1"/>

            <Button Click="PDF_Click" Content="PDF" HorizontalAlignment="Left"
Margin="442,270,0,0" VerticalAlignment="Top" Width="75"
RenderTransformOrigin="0.72,0.5" Grid.Column="1"/>

            <Button Content="Salvar dados" HorizontalAlignment="Left"
Margin="16,206,0,0" VerticalAlignment="Top" Width="75"
Click="Salvardados_Click" Grid.Column="1"/>

            <DataGrid
PreviewMouseDoubleClick="gridpagamento_PreviewMouseDoubleClick"
Name="gridpagamento" HorizontalAlignment="Left" Height="131"
Margin="134,128,0,0" VerticalAlignment="Top" Width="383"
SelectionChanged="DataGrid_SelectionChanged" Grid.Column="1">

                <DataGrid.Columns>

                    <!--<DataGridTextColumn Header="ID" Binding="{Binding id}" />

                    <DataGridTextColumn Header="Nome" Binding="{Binding Nome}"
/>

                    <DataGridTextColumn Header="Morada" Binding="{Binding
Morada}"/>

                    <DataGridTextColumn Header="Localidade" Binding="{Binding
Localidade}"/>

                    <DataGridTextColumn Header="CC" Binding="{Binding CC}"/>

                    <DataGridTextColumn Header="NIF" Binding="{Binding NIF}"/>
```

```
                <DataGridTextColumn Header="Foto" Binding="{Binding Foto}"/>
            </DataGrid.Columns>
        </DataGrid>
        <TextBlock  HorizontalAlignment="Left" Margin="356,269,0,0"
TextWrapping="Wrap" VerticalAlignment="Top" Width="71" Text="Emitir fatura:"
Grid.Column="1"/>
        <TextBlock  HorizontalAlignment="Left" Margin="1,10,0,0"
TextWrapping="Wrap" VerticalAlignment="Top" Text="Dados para pagamento"
Grid.Column="1"/>
        <TextBox Name="tbdelete" HorizontalAlignment="Left" Height="23"
Margin="11,267,0,0" TextWrapping="Wrap" VerticalAlignment="Top" Width="39"
TextChanged="TextBox_TextChanged_2" Grid.ColumnSpan="2"/>
        <TextBlock  HorizontalAlignment="Left" Margin="10,251,0,0"
TextWrapping="Wrap" VerticalAlignment="Top" Grid.ColumnSpan="2"><Run
Text="Eliminar ID"/><Run Text=":"/></TextBlock>
        <Button Click="delete_click" Content="Eliminar"
HorizontalAlignment="Left" Margin="32,267,0,0" VerticalAlignment="Top"
Width="75" Grid.Column="1"/>
        <!--<TextBox Name="txtnum"  HorizontalAlignment="Left" Height="23"
Margin="134,83,0,0" TextWrapping="Wrap" VerticalAlignment="Top" Width="171"
Grid.Column="1"/>-->
        <TextBox Name="txtcaminho" HorizontalAlignment="Left" Height="42"
Margin="134,83,0,0" TextWrapping="Wrap" Text="" VerticalAlignment="Top"
Width="171" Grid.Column="1"/>
        <Button Click="insert_click" Content="Actualizar"
HorizontalAlignment="Left" Margin="118,267,0,0" VerticalAlignment="Top"
Width="75" Grid.Column="1"/>          </Grid>          </TabItem>      </TabControl>
    </Grid>
</Window>
```

Anexo IV

Código Fonte da Janela de Menus

```
using Microsoft.Win32;
using System;
using System.Collections.Generic;
using System.IO;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows;
using System.Windows.Controls;
using System.Windows.Data;
using System.Windows.Documents;
using System.Windows.Input;
using System.Windows.Media;
using System.Windows.Media.Imaging;
using System.Windows.Shapes;
using iTextSharp.text;
using iTextSharp.text.pdf;
using System.Diagnostics;
using System.ComponentModel;
using dadosacesso;
namespace WpfApp1
{
```

```
public partial class Menus : Window
{
    public class veiculo
    {
        public string nome { get; set; }
        public string acessorios { get; set; }
        public int custo { get; set; }
        //public int moeda { get; set; }
    }
    public int valor;
    public List<veiculo> veiculos = new List<veiculo>();
    public Menus()
    {
        InitializeComponent();
        Loadmotas();
    }
    protected void Loadmotas()
    {
        using (Loja1Entities6 db = new Loja1Entities6())
        {
            var qmotas = (from c in db.tbmotas select c).ToList();
            grdListaTudo.ItemsSource = qmotas;
            gridpagamento.ItemsSource = db.Cliente1.ToList();
        }
```

```
}
//Dados dos Botões
private void KTM_Click(object sender, RoutedEventArgs e)
{
   actualiza(4000, "KTM", "");
}
private void Kawasaki_Click(object sender, RoutedEventArgs e)
{
   actualiza(5000, "Kawasaki", "");
}
private void BMW_Click(object sender, RoutedEventArgs e)
{
   actualiza(14000, "BMW", "");
}
private void Suzuki_Click(object sender, RoutedEventArgs e)
{
   actualiza(2000, "Suzuki", "");
}
private void Yamaha_Click(object sender, RoutedEventArgs e)
{
   actualiza(6000, "Yamaha", "");
}
private void Honda_Click(object sender, RoutedEventArgs e)
{
```

```
    actualiza(4000, "Honda", "");
}
private void Button_luvas(object sender, RoutedEventArgs e)
{
    actualiza(120, "", "Luvas Drenaline");
}
private void Button_Capacete(object sender, RoutedEventArgs e)
{
    actualiza(400, " ", "Capacete Nexx Xr2");
}
private void Button_botas(object sender, RoutedEventArgs e)
{
    actualiza(350, "", "Botas Alpinestar");
}
private void Button_Casaco(object sender, RoutedEventArgs e)
{
    actualiza(250, "", "Casaco Thor");
}
private void Button_Camara(object sender, RoutedEventArgs e)
{
    actualiza(500, "", "GoPro Hero");
}
private void Button_viseira(object sender, RoutedEventArgs e)
{
```

```
    actualiza(40, "", "Viseira");
}
private void Button_Click(object sender, RoutedEventArgs e)
{
    this.Close();
    MainWindow mm = new MainWindow();
    mm.Show();
}
private void Button_logout(object sender, RoutedEventArgs e)
{
    this.Close();
    MainWindow mm = new MainWindow();
    mm.Show();
}
private void Pagar_click(object sender, RoutedEventArgs e)
{
    tabpagamento.IsSelected = true;
}
private void Button_Click_1(object sender, RoutedEventArgs e)
{
    tabcarrinho.IsSelected = true;
}
private void Button_Click_2(object sender, RoutedEventArgs e)
{
```

```
    tabacessorios.IsSelected = true;
}
private void Button_Click_3(object sender, RoutedEventArgs e)
{
    tabcarrinho.IsSelected = true;
}
private void calculadora_Click(object sender, RoutedEventArgs e)
{
    Calculadora ss1 = new Calculadora();
    ss1.Show();
}
protected void Motas_Event(object sender, EventArgs args)
{
    Button MotasEvent = sender as Button;
}
//#################################################

void actualiza(int custo, string nome, string acessorios)
{
    using (Loja1Entities6 db = new Loja1Entities6())
    {
        tbmota mymota = new tbmota
        {
            mota = nome,
```

```
            acessorios = acessorios,
            custo = custo,
        };
        db.tbmotas.Add(mymota);
        db.SaveChanges()
        double? valorTotal = db.tbmotas.Sum(s => s.custo);
        totalPrice.Text = valorTotal.ToString();
        grdListaTudo.ItemsSource = db.tbmotas.ToList();

        gridmotas.ItemsSource = db.tbmotas.ToList();
        gridacessorios.ItemsSource = db.tbmotas.ToList();
    protected void grdListaTudo_SelectionChanged(object sender,
SelectionChangedEventArgs e)
    {
    }
    private void Button_remover(object sender, RoutedEventArgs e)
    {
    }
    private void btinsert_Click(object sender, RoutedEventArgs e)
    {
      using (Loja1Entities6 db = new Loja1Entities6())
      {
        if (!int.TryParse(tbvalor.Text, out int valor))
        {
          valor = 0;
```

```
            }
            tbmota mymota = new tbmota
            {
                mota = tbmota.Text,
                acessorios = tbacessorio.Text,
                custo = valor
            };
            db.tbmotas.Add(mymota);
            db.SaveChanges();
            double? valorTotal = db.tbmotas.Sum(s => s.custo);
            totalPrice.Text = valorTotal.ToString();

            grdListaTudo.ItemsSource = db.tbmotas.ToList();
        }
    }
    private void btupdate_Click(object sender, RoutedEventArgs e)
    {
        using (Loja1Entities6 db = new Loja1Entities6())
        {
            int idmota;
            if (int.TryParse(tbidmota.Text, out idmota))
            {
                tbmota mymota = db.tbmotas.Where(x => x.idmota ==
idmota).FirstOrDefault();
                db.SaveChanges();
```

```
                mymota.mota = tbmota.Text;
                mymota.acessorios = tbacessorio.Text;
                mymota.custo = double.Parse(tbvalor.Text);

                double? valorTotal = db.tbmotas.Sum(s => s.custo);
                totalPrice.Text = valorTotal.ToString();

                grdListaTudo.ItemsSource = db.tbmotas.ToList();
            }
        }
    }
    private void btdelete_Click(object sender, RoutedEventArgs e)
    {
        using (Loja1Entities6 db = new Loja1Entities6())
        {
            int idmota;
            if (int.TryParse(tbidmota.Text, out idmota))
            {
                tbmota mymota = db.tbmotas.Where(x => x.idmota ==
idmota).FirstOrDefault();
                if (mymota != null) db.tbmotas.Remove(mymota);
                else MessageBox.Show("Este registo já foi eliminado!");
                db.SaveChanges();
                double? valorTotal = db.tbmotas.Sum(s => s.custo);
                totalPrice.Text = valorTotal.ToString();
```

```
                grdListaTudo.ItemsSource = db.tbmotas.ToList();
            }
        }
    }
    //PAGAMENTOS
    private void Procurar_Click(object sender, RoutedEventArgs e)
    {
        OpenFileDialog dlg = new OpenFileDialog();
        dlg.DefaultExt = ".png";
        dlg.Filter = " *.*|*.*|JPEG Files (*.jpeg)|*.jpeg|PNG Files (*.png)|*.png|JPG
Files (*.jpg)|*.jpg|GIF Files (*.gif)|*.gif";
        dlg.Title = "Foto";
        //dlg.InitialDirectory = pasta;
        Nullable<bool> result = dlg.ShowDialog();
        // Get the selected file name and display in a TextBox
        //if (result == true && !String.IsNullOrEmpty(txtnum.Text))
        if (result == true && dlg.FileNames[0].Length > 0)
        {
            // Open document
            //string fich = txtnum.Text + System.IO.Path.GetExtension(dlg.FileName);
            //txtcaminho.Text = utils.caminhoAppSubPasta("imagens", fich);

            txtcaminho.Text = dlg.FileName;
            imagem.Source = utils.picpelocaminho(dlg.FileName);
            //utils.SaveUsingEncoder(img, utils.caminhoAppSubPasta("imagens", fich));
```

```
    }
}
public string GetFileNameNoExt(string FilePathFileName)
{
    return System.IO.Path.GetFileNameWithoutExtension(FilePathFileName);
}
private void Salvardados_Click(object sender, RoutedEventArgs e)
{
    using (Loja1Entities6 db = new Loja1Entities6())
    {
        byte[] image = null;
        if (!string.IsNullOrEmpty(txtcaminho.Text))
        {
            image = utils.ConvertBitmapSourceToByteArray(txtcaminho.Text);
        }
        Cliente1 mycliente = new Cliente1
        {
            Nome = Tbnome.Text,
            Morada = tbmorada.Text,
            Localidade = tblocalidade.Text,
            NIF = tbnif.Text,
            CC = tbcc.Text,
            Foto = image
        };
```

```
            db.Cliente1.Add(mycliente);
            db.SaveChanges();
            gridpagamento.ItemsSource = db.Cliente1.ToList();
        }
    }
    private void PDF_Click(object sender, RoutedEventArgs e)
    {
        //newstrembuilder.add(srting) ao fim mandar para tostring
        Document doc = new Document(PageSize.A4);
        string dir = @"C:\PDF" + @"\pagamento.pdf";
        var output = new FileStream(dir, FileMode.Create);
        var writer = PdfWriter.GetInstance(doc, output);
        doc.Open();

        //gridpagamento.ItemsSource = db.cliente1.ToList();
        PdfPTable table = new PdfPTable(7);
        table.DefaultCell.BorderColor = BaseColor.BLACK;
        table.DefaultCell.HorizontalAlignment = Element.ALIGN_CENTER;
        PdfPCell cell = new PdfPCell();
        cell.Colspan = 7;
        cell.HorizontalAlignment = Element.ALIGN_CENTER;
        cell.VerticalAlignment = Element.ALIGN_CENTER;
        table.AddCell(cell);
        table.AddCell(new Phrase("Id"));
```

```
table.AddCell(new Phrase("Nome"));
table.AddCell(new Phrase("Morada"));
table.AddCell(new Phrase("Localidade"));
table.AddCell(new Phrase("CC"));
table.AddCell(new Phrase("NIF"));
table.AddCell(new Phrase("Foto"));
using (Loja1Entities6 db = new Loja1Entities6())
{
   foreach (Cliente1 i in db.Cliente1)
   {
      table.AddCell(new Phrase(i.id.ToString()));
      table.AddCell(new Phrase(i.Nome.ToString()));
      table.AddCell(new Phrase(i.Morada.ToString()));
      table.AddCell(new Phrase(i.Localidade.ToString()));
      table.AddCell(new Phrase(i.NIF.ToString()));
      table.AddCell(new Phrase(i.CC.ToString()));
      //var foto = utils.ConvertByteArrayToBitmapImage(i.Foto);
      var foto = iTextSharp.text.Image.GetInstance(i.Foto);
      foto.ScaleAbsoluteHeight(200);
      foto.ScaleAbsoluteWidth(170);
      table.AddCell(foto);
   }
}
doc.Add(table);
```

```
doc.Add(new Phrase("\n\n"));
PdfPTable table1 = new PdfPTable(5);
table1.DefaultCell.BorderColor = BaseColor.BLACK;
table1.DefaultCell.HorizontalAlignment = Element.ALIGN_CENTER;
PdfPCell cell1 = new PdfPCell();
cell1.Colspan = 5;
cell1.HorizontalAlignment = Element.ALIGN_CENTER;
cell1.VerticalAlignment = Element.ALIGN_CENTER;
table1.AddCell(cell1);
table1.AddCell(new Phrase("idmota"));
table1.AddCell(new Phrase("Mota"));
table1.AddCell(new Phrase("Acessorios"));
table1.AddCell(new Phrase("Valor"));
table1.AddCell(new Phrase("c/Iva"));
using (Loja1Entities6 db = new Loja1Entities6())
{
  foreach (tbmota i in db.tbmotas)
  {
    table1.AddCell(new Phrase(i.idmota.ToString()));
    table1.AddCell(new Phrase(i.mota.ToString()));
    table1.AddCell(new Phrase(i.acessorios.ToString()));
    table1.AddCell(new Phrase(i.custo.ToString()));
    table1.AddCell(new Phrase(i.custoiva.ToString()));
  }
```

```
            }
            doc.Add(table1);
            doc.Close();
            Process.Start(dir);
        }
        private void delete_click(object sender, RoutedEventArgs e)
        {
            using (Loja1Entities6 db = new Loja1Entities6())
            {
                int id;
                if (int.TryParse(tbdelete.Text, out id))
                {
                    Cliente1 cliente1 = db.Cliente1.Where(x => x.id == id).FirstOrDefault();
                    if (cliente1 != null) db.Cliente1.Remove(cliente1);
                    else MessageBox.Show("Este registo já foi eliminado!");
                    db.SaveChanges();
                    gridpagamento.ItemsSource = db.Cliente1.ToList();
                    tbdelete.Text = "";
                }
            }
        }
        private void gridpagamento_PreviewMouseDoubleClick(object sender,
MouseButtonEventArgs e)
        {
            gridpagamento.IsReadOnly = true;
```

```
        DataGrid d = (DataGrid)sender;
        int indexRow = d.SelectedIndex;

        var id =
((WpfApp1.Cliente1)((System.Windows.Controls.ItemsControl)sender).Items[indexRo
w]).id;
        using (Loja1Entities6 db = new Loja1Entities6())
        {
            var cliente = db.Cliente1.Where(w => w.id == id).FirstOrDefault();
            tbdelete.Text = cliente.id.ToString();
            Tbnome.Text = cliente.Nome;
            tbmorada.Text = cliente.Morada;
            tblocalidade.Text = cliente.Localidade;
            tbnif.Text = cliente.NIF;
            tbcc.Text = cliente.CC;
            if (cliente.Foto != null)
            {
                imagem.Source = utils.ConvertByteArrayToBitmapImage(cliente.Foto);
            }
        }
    }
    private void insert_click(object sender, RoutedEventArgs e)
    {

        using (Loja1Entities6 db = new Loja1Entities6())
```

```
        {
            int.TryParse(tbdelete.Text, out int a);
            var mycliente = db.Cliente1.Where(w => w.id == a).FirstOrDefault();
            mycliente.Nome = Tbnome.Text.ToString();
            mycliente.Morada = tbmorada.Text.ToString();
            mycliente.Localidade = tblocalidade.Text.ToString();
            mycliente.NIF = tbnif.Text.ToString();
            mycliente.CC = tbcc.Text.ToString();
            if (!string.IsNullOrEmpty(txtcaminho.Text))
            {
                mycliente.Foto =
utils.ConvertBitmapSourceToByteArray(txtcaminho.Text);
            }
            db.SaveChanges();
            gridpagamento.ItemsSource = db.Cliente1.ToList();
        }
    }
```

Anexo V

XAML da Janela da Calculadora

```
<Window x:Class="WpfApp1.Calculadora"
        xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
        xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
        xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
        xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
        xmlns:local="clr-namespace:WpfApp1"
        mc:Ignorable="d"
        Title="Calculadora" Height="266.912" Width="317.647" Background="Black"
Icon="C:\Barrigas 2\wpfapp1\WpfApp1\Imagens\ico.png">
    <Canvas Margin="0,0,-8,-21">
        <TextBlock TextAlignment="Right" FontSize="30" x:Name="visor" Width="290"
Height="50" Background="White" Canvas.Left="8" Canvas.Top="10"></TextBlock>
```

```
        <WrapPanel HorizontalAlignment="Left" Height="157"
VerticalAlignment="Top" Width="171" Canvas.Left="19" Canvas.Top="68" >
            <Button x:Name="bt1" Content="1" Width="46" Height="34" Margin="2"
Click="btnum_Click"/>
            <Button x:Name="bt2" Content="2" Width="46" Height="34" Margin="2"
Click="btnum_Click"/>
            <Button x:Name="bt3" Content="3" Width="46" Height="34" Margin="2"
Click="btnum_Click"/>
            <Button x:Name="bt4" Content="4" Width="46" Height="34" Margin="2"
Click="btnum_Click"/>
            <Button x:Name="bt5" Content="5" Width="46" Height="34" Margin="2"
Click="btnum_Click"/>
            <Button x:Name="bt6" Content="6" Width="46" Height="34" Margin="2"
Click="btnum_Click"/>
            <Button x:Name="bt7" Content="7" Width="46" Height="34" Margin="2"
Click="btnum_Click"/>
            <Button x:Name="bt8" Content="8" Width="46" Height="34" Margin="2"
Click="btnum_Click"/>
            <Button x:Name="bt9" Content="9" Width="46" Height="34" Margin="2"
Click="btnum_Click"/>
            <Button x:Name="bt0" Content="0" Width="46" Height="34" Margin="2"
Click="btnum_Click"/>
            <Button x:Name="btponto" Content="," Width="46" Height="34"
Margin="2" Click="btnum_Click"/>
            <Button x:Name="btnega" Content="-" Width="46" Height="34" Margin="2"
Click="btnega_Click"/>
        </WrapPanel>
        <WrapPanel Height="157" Canvas.Left="177" Canvas.Top="68" Width="100">
            <Button x:Name="btmult" Content="*" Width="46" Height="34" Margin="2"
Click="btopera_Click" />
            <Button x:Name="btsoma" Content="+" Width="46" Height="34" Margin="2"
Click="btopera_Click" />
            <Button x:Name="btdivid" Content="/" Width="46" Height="34"
Margin="2" Click="btopera_Click"/>
            <Button x:Name="btsub" Content="-" Width="46" Height="34" Margin="2"
Click="btopera_Click"/>
            <Button x:Name="btC" Content="C" Width="46" Height="34" Margin="2"
Click="btCLEAN_Click"/>
                        <Button x:Name="btigual" Content="=" Width="97"
Height="34" Margin="2" Click="btigual_Click"/>
        </WrapPanel>

    </Canvas>
</Window>
```

Anexo VI

Código Fonte da Janela de Autenticação

```
using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows;
```

```
using System.Windows.Controls;
using System.Windows.Data;
using System.Windows.Documents;
using System.Windows.Input;
using System.Windows.Media;
using System.Windows.Media.Imaging;
using System.Windows.Shapes;
namespace WpfApp1
{
    public delegate float? dlgopera(float? a, float? b);
    public partial class Calculadora : Window
    {
        bool limpaVisor = true;
        bool primeiroArg = true;
        float? arg1, arg2;
        dlgopera func = null;
        public Calculadora()
        {
            InitializeComponent();
        }
        private void btnum_Click(object sender, RoutedEventArgs e)
        {
            Button bt = (Button)sender;
            if (limpaVisor) { visor.Text = ""; limpaVisor = false; }
            visor.Text += bt.Content.ToString();
        }
        private void btCLEAN_Click(object sender, RoutedEventArgs e)
        {
            visor.Text = "";
            arg1 = null;
```

```
        arg2 = null;
        limpaVisor = true;
        primeiroArg = true;
        func = null;
    }
    private void btigual_Click(object sender, RoutedEventArgs e)
    {
        try
        {
            if(arg1 !=null && !string.IsNullOrEmpty(visor.Text)&& func !=null)
            {
                arg2 = float.Parse(visor.Text);
                visor.Text = func(arg1, arg2).ToString();
                arg2 = arg2 = null;
                func = null;
                limpaVisor = true;
                primeiroArg = true;
            }
        }
        catch (Exception erro)
        {
            visor.Text = erro.Message;
        }
    }
    private void btopera_Click(object sender, RoutedEventArgs e)
    {
        Button bt = (Button)sender;
        try
        {
            if(primeiroArg)
```

```
{
    arg1 = float.Parse(visor.Text);
    primeiroArg = false;
    limpaVisor = true;
}else
{
    if(func !=null && !string.IsNullOrEmpty(visor.Text))
    {
        arg2 = float.Parse(visor.Text);
        arg1 = func(arg1, arg2);
        visor.Text = arg1.ToString();
        arg2 = null;
        limpaVisor = true;

    }
}

switch(bt.Content.ToString())
{
    case "+":
        func = (a, b) => a + b;
        break;
    case "-":
        func = (a, b) => a - b;
        break;
    case "/":
        func = (a, b) => a / b;
        break;
    case "*":
        func = (a, b) => a * b;
```

```
                break;
            }
        }
        catch (Exception erro)
        {
            visor.Text = erro.Message;
        }
    }
    private void btnega_Click(object sender, RoutedEventArgs e)
    {
        try
        {
            float? rslt = float.Parse(visor.Text);
            rslt *= -1;
            visor.Text = rslt.ToString();
        }
        catch (Exception erro)
        {
            visor.Text = erro.Message;
        }
    }
  }
}
```