



Projeto Global

Licenciatura em Informática

Turma B

Tiago Costa Nº 8147

Orientador: Professor Pedro Brandão

Lisboa

Ano Letivo: 2017/2018

Resumo

Este projeto tem como objetivo a criação de um laboratório de virtualização de servidores com diversas funções e características, com auxílio das tecnologias apropriadas, que permite a simulação de um ambiente empresarial.

As tecnologias referentes ao projeto, serão abordadas em relação ao impacto que têm na sua área, à sua descrição (o que são, o que fazem), às vantagens e desvantagens das suas implementações, aos tipos de implementação, às suas características e às suas funções.

Com a implementação do laboratório pode-se verificar os benefícios que as tecnologias usadas trazem, redução de custos a nível financeiro, de manutenção e de espaços físicos.

Palavras-chave: Virtualização, Servidores, Tecnologias, Simulação, Vantagens, Desvantagens, Características, Funções.

Abstract

This project focuses on the creation of a server virtualization lab with various functions and characteristics, with the help of appropriate technologies, which allows the simulation of a business environment.

The technologies related to the project, will be approached in relation to the impact they have in their area, their description (what they are, what they do), the advantages and disadvantages of their implementations, the types of implementation, their characteristics and their functions.

With the implementation of the laboratory, one can verify the benefits that the technologies used bring, such as, reduction of costs in financial, maintenance and physical spaces.

Keywords: Virtualization, Servers, Technologies, Simulation, Advantages, Disadvantages, Features, Functions.

Abreviaturas

DNS - Domain Name System

EAP - Extensible Authentication Protocol

IP – Internet Protocol

IT - Information Technology

NIC - Network interface controller

SQL - Structured Query Language

TTLS - Tunneled Transport Layer Security

WPF - Windows Presentation Foundation

Índice

Resumo.....	i
Abstract	ii
Abreviatura.....	iii
Índice de figuras	v
Introdução	1
Estado de arte	2
Contextualização	7
Desenvolvimento.....	8
Conclusão	51
Referências Bibliográficas	52
Anexos	54

Índice de figuras

Figura I. Tipos de Hypervisor (Webtechmag,2018)	2
Figura II - Configurações das Máquinas Virtuais (do autor)	9
Figura III - Instalação Sistema Operativo (do autor)	10
Figura IV- Instalação Sistema Operativo II (do autor)	10
Figura V - Instalação Sistema Operativo III (do autor)	11
Figura VI - Instalação Sistema Operativo IV (do autor).....	11
Figura VII - Instalação Sistema Operativo V (do autor).....	12
Figura VIII - Instalação Sistema Operativo VI (do autor)	12
Figura IX - Instalação Sistema Operativo VII (do autor)	13
Figura X - Instalação Sistema Operativo VIII (do autor)	13
Figura XI - Configuração da interface de rede da Máquina Virtual DC (do autor).....	14
Figura XII - Nome da Máquina Virtual e nome do Domínio (do autor).....	14
Figura XIII - Configuração do Sufixo do DNS (do autor).....	15
Figura XIV - Windows Firewall Domain: On (do autor)	15
Figura XV - Configuração da interface de rede da Máquina Virtual ROUTING (do autor)...	16
Figura XVI - Configuração da interface de rede da Máquina Virtual SQL (do autor).....	16
Figura XVII - Configuração da interface de rede da Máquina Virtual CLIENT (do autor)....	17
Figura XVIII - Associar Máquina Virtual ao Dominio (do autor).....	18
Figura XIX - Credenciais do Administrador do Dominio (do autor).....	18
Figura XX - Adicionar o Role de Routing (do autor)	19
Figura XXI - Estado do Serviço Routing (do autor)	20
Figura XXII - Instalação do SQL I (do autor).....	20
Figura XXIII - Instalação do SQL II (do autor).....	20
Figura XXIV - Ligar ao servidor SQL (do autor)	21
Figura XXV - Query da criação da base de dados (do autor)	22
Figura XXVI - Query para criar tabela empregados (do autor)	22
Figura XXVII - Query para criar tabela pedidos (do autor).....	23
Figura XXVIII - Adicionar Entity Data Model I (do autor)	24
Figura XXIX - Adicionar Entity Data Model II (do autor).....	24
Figura XXX - Adicionar Entity Data Model III (do autor).....	25
Figura XXXI - Adicionar Entity Data Model IV (do autor)	25
Figura XXXII - Adicionar Entity Data Model V (do autor)	26

Figura XXXIII - Connection String (do autor)	26
Figura XXXIV - Janela Login (do autor).....	26
Figura XXXV - Botão Validar (do autor).....	27
Figura XXXVI - Classe autêntica (do autor)	28
Figura XXXVII - Page1 (do autor)	29
Figura XXXVIII - XAML MainWindow ViewModel (do autor)	29
Figura XXXIX - XAML Conversor Page1 (do autor)	30
Figura XL - XAML Page1 ListView ListEmp (do autor)	30
Figura XLI - ListEmp ViewModel (do autor).....	30
Figura XLII - TextBlocks com o Empregado Corrente (do autor)	30
Figura XLIII - Imagem do Empregado Corrente (do autor)	30
Figura XLIV - TextBox Procura Nome (do autor)	30
Figura XLV - Código para filtrar nome ViewModel (do autor)	31
Figura XLVI - CheckBox Filtrar por Função (do autor).....	31
Figura XLVII - Código para agrupar por Função (do autor)	31
Figura XLVIII - Botão ordenar Id e ordenar nome (do autor)	32
Figura XLIX - Função ordenar Id (do autor)	32
Figura L - Botão Load Foto (do autor).....	32
Figura LI - Classe Comandos (do autor).....	33
Figura LII - CommandBiding Page1 (do autor).....	33
Figura LIII - Código para atualizar a imagem do empregado corrente (do autor).....	34
Figura LIV - Botões Download e Download Todos (do autor)	34
Figura LV - Botões para navegar na ListView (do autor)	37
Figura LVI - Botão Update (do autor)	38
Figura LVII - Botão Delete (do autor)	39
Figura LVIII - Botão Insert (do autor)	40
Figura LIX - Página dois (do autor).....	41
Figura LX - Botão Carregar Foto (do autor).....	42
Figura LXI - Botão Inserir (do autor)	42
Figura LXII - Page3 (do autor)	44
Figura LXIII - XAML Page3 ViewModelPedidos (do autor)	45
Figura LXIV - ListView empregados (do autor)	45
Figura LXV – Listar pedidos do empregado (do autor).....	45
Figura LXVI - Função apagarpedido (do autor)	47

Figura LXVII - Função updatepedido (do autor).....	48
Figura LXVIII - Filtrar o nome (do autor)	54
Figura LXIX - Agrupar por função (do autor)	54
Figura LXX - Ordenar Id descendente (do autor).....	55
Figura LXXI - Ordenar nome ascendente (do autor)	55
Figura LXXII - Ordenar nome descendente (do autor).....	55
Figura LXXIII - Pdf individual (do autor)	56

Introdução

A virtualização permite executar serviços, programas ou sistemas operativos, possibilitando, inclusive, simular *hardwares* diferentes, tais como interfaces de rede, *switchs*, entre outros, o que pode original uma plataforma *cloud*. Atualmente, a virtualização seja ela de serviços, aplicações ou de servidores é usada em muitas áreas, relacionadas diretamente com as tecnologias ou não.

Este projeto tem como objetivo simular um ambiente empresarial em pequena escala, com quatro máquinas virtuais com o sistema operativo Windows Server 2012 R2, um controlador de domínio, um servidor de *routing*, um servidor SQL e um cliente com o auxílio do *software* VMWare Workstation Pro 12. A aplicação criada no Visual Studio serve para gerir os empregados e os pedidos de uma empresa e o acesso é restrito a utilizadores do domínio.

Este relatório inclui todos os procedimentos realizados na elaboração do laboratório, contidos no capítulo de Desenvolvimento. O capítulo de Estado de Arte aborda detalhadamente todas as tecnologias usadas na elaboração deste projeto.

Estado de arte

A virtualização é um processo que permite criar e executar vários ambientes virtuais baseados em aplicações, sistemas operativos, armazenamento e redes, sobre um servidor físico. Para tal terá de partilhar os recursos do servidor físico com os ambientes virtuais (máquinas virtuais). (Open Source, 2018)

Em relação ao IT convencional, a virtualização permite a redução dos custos a nível de *hardware*, do consumo de energia, maior desempenho e disponibilidade de recursos, melhor capacidade de recuperação, ser escalável e melhor gestão. (VMware, 2018)

O servidor físico tem de correr um *hypervisor*. Um *hypervisor* é um programa que permite criar e correr máquinas virtuais. Existem dois tipos de *hypervisor*, tipo um (*bare metal*) e tipo dois (*hosted*). Tipo um corre diretamente no *hardware* do servidor, o *hypervisor* pode ser considerado o sistema operativo. Tipo dois funciona como uma aplicação, ou seja, o *hypervisor* está instalado num sistema operativo, mas não trabalha diretamente com o *hardware* do servidor, pois ainda tem de passar pelo sistema operativo até chegar ao *hardware*. (Red Hat, 2018)

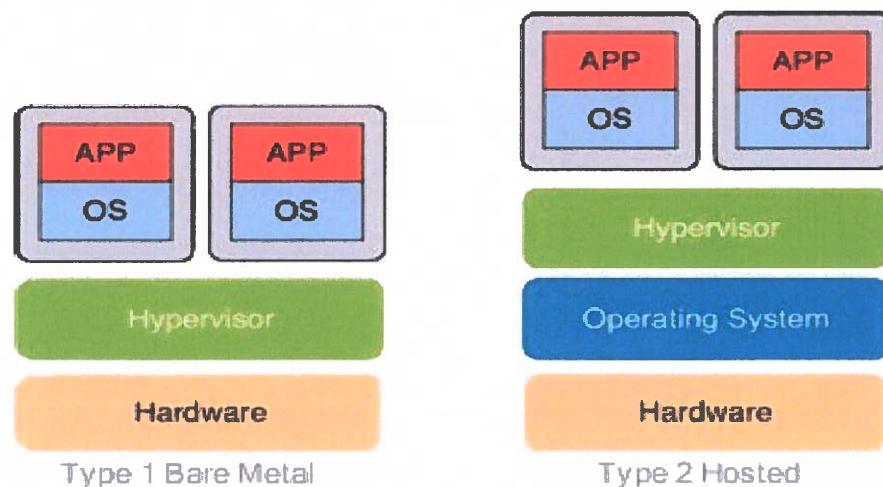


Figura I. Tipos de Hypervisor

Fonte: (Webtechmag, 2018).

Uma máquina virtual é o equivalente a um servidor ou computador físico, mas em *software*, ou seja, é um ficheiro que pode correr numa máquina física. Seria como se instalasse um computador dentro de outro, o que torna possível ter várias máquinas virtuais no mesmo servidor físico, o *hypervisor* é responsável pela criação e gestão das mesmas. Cada máquina virtual tem o seu *hardware* virtual (processadores, memória, armazenamento, interfaces de rede, entre outros), que é mapeado com o *hardware* da máquina física. (Microsoft, 2018)

O Windows Server 2012 R2 é um sistema operativo, que permite ter uma plataforma de virtualização e computação em *cloud* de alto nível, que em junção com Windows Azure e o System Center será uma plataforma altamente escalável para aplicações e oferece um suporte melhorado para padrões abertos. Também facilita a implementação de uma infraestrutura de *desktop virtual* (VDI). Esta infraestrutura disponibiliza aos utilizadores acesso a recursos de IT (*information technology*) em qualquer lugar, garantindo sempre a segurança dos dados. (Tulloch, 2013, pp. 1-3)

O Windows Server 2012 R2 trouxe inovações em relação aos seus predecessores em alguns níveis. A nível do Active Directory Domain Services (ADDS) tem o Active Directory Recycle Bin, que permite recuperar objetos que são apagados no Active Directory. O Active Directory Administrative Center (ADAC), é interface orientada para gestão do Active Directory, que permite a ligação a um ou mais domínios, possibilita a criação e gestão de contas de utilizador, grupos, unidades organizacionais e também pesquisas globais no Active Directory. O Cloning Domain Controllers permite clonar um Controlador de Domínio, para tal, o Controlador de Domínio tem que ser virtualizado por um hypervisor (ex: Hyper-V). O Fine-Grained Password Policy Improvements, permite a partir do Active Directory Administrative Center criar e administrar os nossos Password Settings Object(PSO). A nível de Rede tem o IP Address Management, que permite administrar, monitorizar e auditar o espaço do endereço IP, permitindo localizar servidores de endereçamento na rede de modo a administra-los a partir de uma interface de utilizador. O NIC Teaming é uma tecnologia que permite agregar várias interfaces de rede numa só. Neste caso permite redundância em caso de falha de uma das interfaces e o balanceamento de rede. O EAP-TTLS é um novo protocolo de autenticação que previne os acessos não autorizados na intranet. O DNS permite ser instalado e removido via Powershell. (Minasi, et al., 2013, pp. 2-17)

O Hyper-V é uma tecnologia que fornece recursos informáticos através da virtualização. Permite criar uma versão de *software* do computador, denominada máquina virtual, utilizada para executar um sistema operativo e aplicações. Pode executar várias máquinas virtuais ao mesmo tempo e pode criá-las e eliminá-las conforme necessário. No Windows Server 2012 R2 o Hyper-V é uma função(*role*). Algumas funções que o Hyper-V permite são, o Live Migration, efetua a migração de máquinas virtuais de um servidor físico para outro, sem impacto na disponibilidade da máquina para os utilizadores. O Cluster Shared Volumes, permite que múltiplos nós de *cluster* tenham simultaneamente acesso de leitura e escrita para o mesmo disco. A compatibilidade do processador aumenta a flexibilidade para o Live Migration entre *hosts* (servidor com um *hypervisor* instalado) com diferentes arquiteturas de CPU. A memória dinâmica é um recurso compartilhado que pode ser redistribuído pelas máquinas virtuais. (*ibid*, pp. 8-10)

O Active Directory permite aos administradores fazer a gestão de informações com eficiência de um repositório central que pode ser distribuído globalmente. Após as informações sobre utilizadores, grupos, computadores, impressoras, aplicativos e serviços forem adicionados ao Active Directory, elas poderão ser disponibilizadas para uso na organização, para quantas pessoas quiser. A estrutura das informações pode corresponder à estrutura da sua organização e dos seus utilizadores podem consultar o Active Directory para encontrar o local de uma impressora ou o endereço de *email* de um colega. Com unidades organizacionais, pode-se delegar controlo e gestão dos dados. (Desmond, Richards, Allen, & Lowe-Norris, 2013, pp. 1-3)

Active Directory é uma base de dados centralizada onde se armazena e gera objetos (utilizadores e computadores) de uma forma hierárquica, é uma função(*role*) do Windows Server 2012 R2. Quando se instala essa função no sistema promove-se um servidor a controlador de domínio, para um domínio existente ou um novo. Cada controlador de domínio tem uma cópia da base de dados do Active Directory, que é atualizada por outros controladores de domínio. Um domínio é uma coleção de objetos que partilham a mesma base de dados, o que permite criar um utilizador no Active Directory e esse poder entrar em qualquer computador que esteja no domínio. Um servidor que executa o Active Directory Domain Services é um controlador de domínio. (Minasi, et al., 2013, pp. 257-260)

MVVM (*Model View ViewModel*) é um padrão de arquitetura baseado em MVC (*Model View-Controller*) e MVP (*Model-View-Presenter*), que tenta separar mais o desenvolvimento de interfaces de utilizador, do comportamento e da lógica de negócios na implementação de uma aplicação. Para este fim, muitas implementações deste padrão fazem uso de *data bindings* que permitem a separação do trabalho em *Views* de outras camadas. Isto facilita o trabalho de desenvolvimento e a interface do utilizador. (MVVM, 2012)

MVVM foi projetado para fazer uso de funções específicas no WPF para facilitar melhor a separação do desenvolvimento da camada *View* do restante padrão, removendo praticamente todos os *code-behinds* desta camada. Em vez de exigir que os designers interativos gravem o código *View*, estes podes usar a linguagem XAML e criar ligações com a classe *ViewModel*, que é escrita e guardada pelos programadores de aplicações. Esta separação de funções permite que os designers se concentrem em certas necessidades de UX (*User eXperience*) do que na lógica de programação ou de negócios, permitindo que as camadas de uma aplicação seja desenvolvida em vários fluxos de trabalho. (Migliore, 2011)

SQL (*Structured Query Language*) é uma língua de programação, usada para fazer a gestão de bases de dados relacionais e executar várias operações com os dados. Para fazer a gestão das bases de

dados, são usados Sistemas de Gestão de Base de Dados (SGBD). Um SGBD é um programa que permite criar, modificar e partilhar as bases de dados entre utilizadores e aplicações. SQL Server 2014 é um SGBD, que fornece uma plataforma consistente para infraestruturas, aplicações e dados que abrangem os *datacenters* do cliente, *datacenter* de fornecedores de serviços de hospedagem e a *cloud* pública da Microsoft. Com o SQL Server 2014, os clientes vão usar uma plataforma que contém o desenvolvimento, a administração, identidade e virtualização independentemente de onde estejam as aplicações. (Mistry & Misner, 2013, pp. 3-6)

O Microsoft Visual Studio é *um IDE (integrated development environment)* da Microsoft.

IDE é um programa que possui várias ferramentas para desenvolver aplicações e serviços *web*.
(Visual Studio, 2018)

A *Entity Framework* foi lançada em 2008, é um *Object Relational Mapper (ORM)*. Um ORM é uma ferramenta que simplifica o mapeamento entre objetos código para as tabelas e colunas de uma base de dados relacional, cuida da criação de ligações com a base de dados e da execução de comandos, além de obter resultados de consulta e materializar automaticamente esses resultados como seus objetos da aplicação. O *Entity Framework* é uma estrutura de código aberto para o ADO.NET, que faz parte do .NET Framework. (Entiry Framework, 2018)

Cloud Computing é a capacidade de computação de alta disponibilidade, escalabilidade e flexibilidade. A *cloud* faz com que a largura de banda, o espaço de armazenamento, a segurança, o poder de processamento não sejam um problema para o consumidor. (Cloud Computing, 2017)

Cloud Computing é um modelo que faz com que seja possível o acesso a uma *pool* partilhada de recursos configurados computacionais, que são fornecidos e removidos com pouco esforço ou comunicação com o fornecedor do serviço, de forma ubíqua, conveniente e *on-demand*. De acordo o modelo 5-3-4 do NIST, existem 5 características essenciais, 3 modelos de serviço e 4 modelos de implementação. As características essenciais são o *Self-service* a pedido, permite ao cliente provisionar capacidades computacionais, sem ter a necessidade de interagir com o fornecedor do serviço. Acesso abrangente à rede, ou seja, os serviços estão disponíveis através da rede que podem ser acedidos por mecanismos padrão que promovem as plataformas heterogéneas. Recursos partilhados computacionais dos fornecedores são agrupados para partilhar por diversos consumidores ao usar um modelo *multi-tenant*, com diferentes recursos físicos e virtuais atribuídos e reatribuídos de acordo com a exigência do cliente. Um cliente não tem controlo ou conhecimento da exata localização dos recursos fornecidos. Elasticidade rápida, permite com que os serviços possam ser rapidamente provisionados e lançados,

em alguns casos automaticamente, para escalar rapidamente para fora e para dentro proporcional à necessidade. Serviço medido, deixa os sistemas *cloud* controlar e otimizar automaticamente o uso de recursos através de influenciar a capacidade de medição ao nível de abstração apropriada ao tipo de serviço. Os recursos podem ser monitorizados, controlados com transparência para ambos o cliente e o fornecedor do serviço. (Mell & Grance, 2011)

Os modelos de Serviço são o SaaS (*Software as a Service*), as aplicações são fornecidas pelo fornecedor do serviço. O cliente não gera nem controla a estrutura *cloud* ou as capacidades da aplicação. Não é viável para aplicações de tempo real, pois o consumidor não tem controlo sobre a estrutura *cloud*. Inclui serviços empresariais e aplicações *Web*. O PaaS (*Platform as a Service*), o cliente controla as aplicações lançadas e possivelmente as configurações de hospedagem, ou seja, o cliente faz o papel de fornecedor de SaaS. O cliente não gera nem controla a estrutura *cloud*. Não é útil quando a aplicação tem de ser portátil, o *hardware* ou o *software* tem que ser *custom performance* ou quando são linguagens de programação proprietárias. O IaaS (*Infrastructure as a Service*) – O utilizador é capaz de lançar e correr software arbitrário, que pode incluir sistemas operativos e aplicações. O cliente não gera nem controla a estrutura *cloud*. O IaaS inclui serviços de *server hosting*, armazenamento, *hardware* e sistemas operativos. (Azure Cloud, 2018)

Os modelos de Implementação são a *cloud* privada, é uma estrutura de *cloud* que serve para o uso exclusivo de uma organização. Pode ser detida, gerida e operada por terceiros. A estrutura não necessita de estar na empresa. A *cloud* comunitária é uma estrutura de *cloud* que serve para o uso específico de uma comunidade de consumidores. Pode ser detida, gerida e operada por uma ou mais organizações das comunidades ou por terceiros. A *cloud* publica é uma estrutura de *cloud* que serve para o uso do público geral. Pode ser detida, gerida e operada por um negócio académico, organização governamental ou uma combinação deles. A estrutura está nas instalações do fornecedor do serviço. *cloud* híbrida é uma estrutura de *cloud* composta por duas ou mais estruturas de *cloud*. São entidades únicas, mas estão ligadas por tecnologia padrão ou proprietária que permite a portabilidade de dados e aplicações. (IBM, 2018)

Contextualização

Este trabalho teve como propósito a exploração de algumas das tecnologias mais recentes no domínio da virtualização, pois seja virtualização de serviços, aplicações ou de servidores, hoje já deixou de ser uma tendência para ser uma realidade em muitos setores, seja dentro ou fora da área da tecnologia.

Hoje em dia as empresas necessitam de ter escalabilidade e elasticidade para que os serviços consigam acompanhar os pedidos e evitem o desperdício na utilização dos recursos computacionais. Neste caso, ter uma infraestrutura dimensionada adequadamente e equipamentos aferidos de forma precisa contribuem para obter excelentes resultados.

O departamento de IT de uma empresa ganha capacidade na entrega dos recursos ao aderir à virtualização, diminui a complexidade e aumenta a flexibilidade da infraestrutura, acumulando vantagens para a empresa.

No caso de uma falha ou desastre do sistema, a virtualização permite uma recuperação mais rápida dos recursos de IT. As infraestruturas mais antigas são incapazes de se recuperar dentro de algumas horas e, na maioria dos casos, as empresas experimentam um tempo de inatividade muito mais longo.

Os ambientes virtualizados são projetados para serem escalonáveis, o que permite mais flexibilidade no que diz respeito ao crescimento da empresa. Em vez de adquirir componentes de infraestrutura adicionais, novas aplicações e atualizações podem ser facilmente implementados com a virtualização.

As empresas conseguem economizar nos custos da infraestrutura informática. A redução de custos também se estende à redução do consumo de energia e da equipa de IT.

Desenvolvimento

Para o desenvolvimento do projeto foi necessário o computador físico preencher um conjunto de requisitos para permitir a criação de várias máquinas virtuais. O computador físico possui as seguintes características:

- Processador – Intel Core i5-254M CPU@ 2.60 GHz
- Memoria RAM – 16 GB
- Armazenamento – 500 GB SSD
- Sistema Operativo – Windows 10
- Hypervisor – VMWare Workstation Pro 12

Para este laboratório foram criadas quatro máquinas virtuais, DC, *ROUTING*, SQL e *CLIENT*.

	Sistema Operativo	Disco	RAM	IP
DC	Windows Server 2012 R2	60 GB	2GB	10.1.1.2
ROUTING	Windows Server 2012 R2	60 GB	2 GB	10.1.1.1
SQL	Windows Server 2012 R2	60 GB	2 GB	10.1.1.3
CLIENT	Windows 8.1	60 GB	4GB	10.1.1.5

Tabela I- Esquema do Laboratório

Fonte: (do autor)

Para a criação das máquinas virtuais, foi usado o VMware Workstation Pro 12.

A configurações das máquinas estão ilustradas na seguinte imagem:

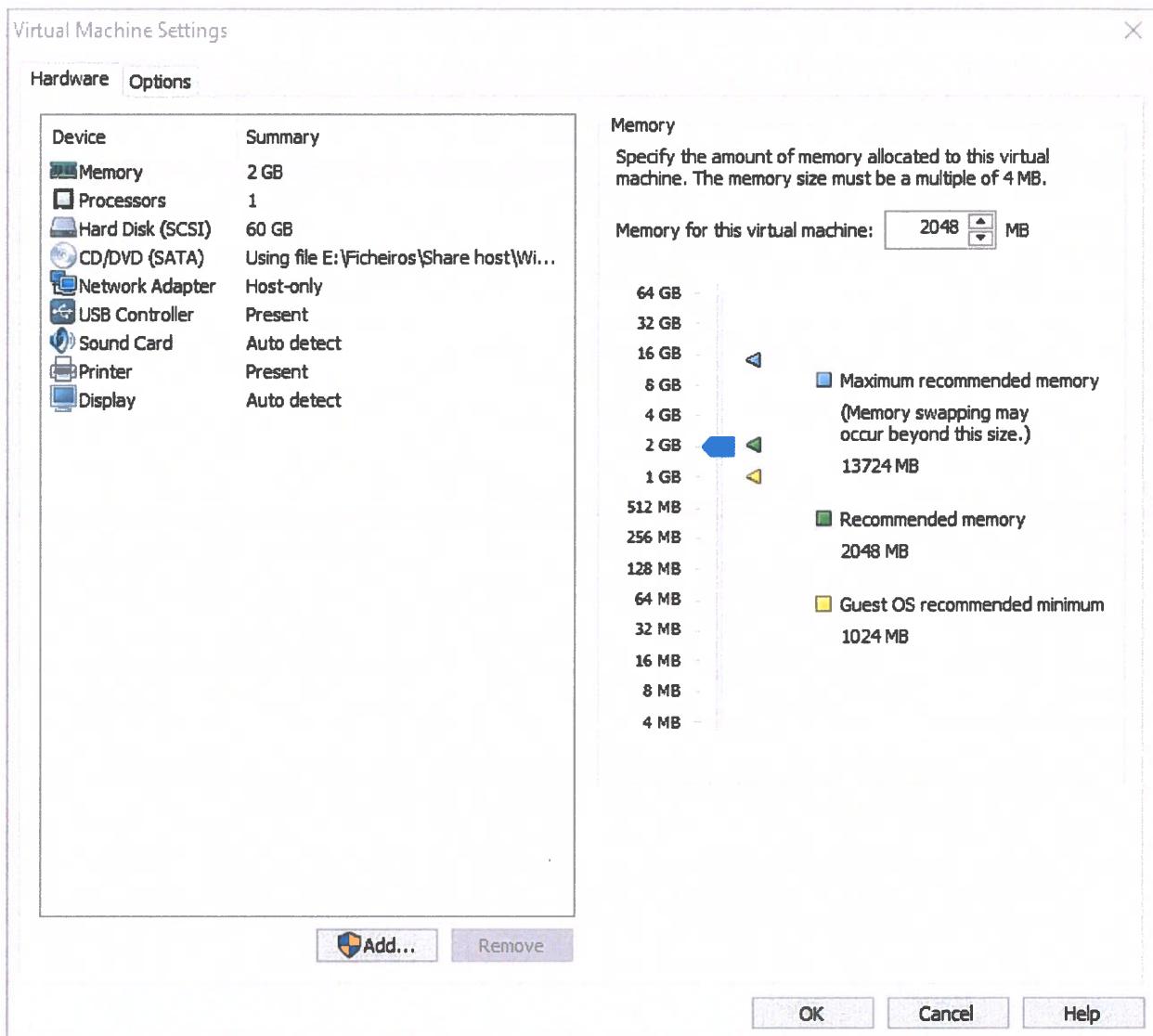


Figura II - Configurações das Máquinas Virtuais

Fonte: (do autor)

A máquina *ROUTING* tem mais uma interface NAT e a máquina *CLIENT* tem 4096 MB de memória RAM.

Após a criação das máquinas, segue-se a instalação do sistema operativo. Em todas as máquinas exceto a *CLIENT* terão o Windows Server 2012 R2, na *CLIENT* tem o Windows 8.1 Multiple Versions.

O processo de instalação das máquinas virtuais está replicado nas seguintes imagens.



Figura III - Instalação Sistema Operativo

Fonte: (do autor)

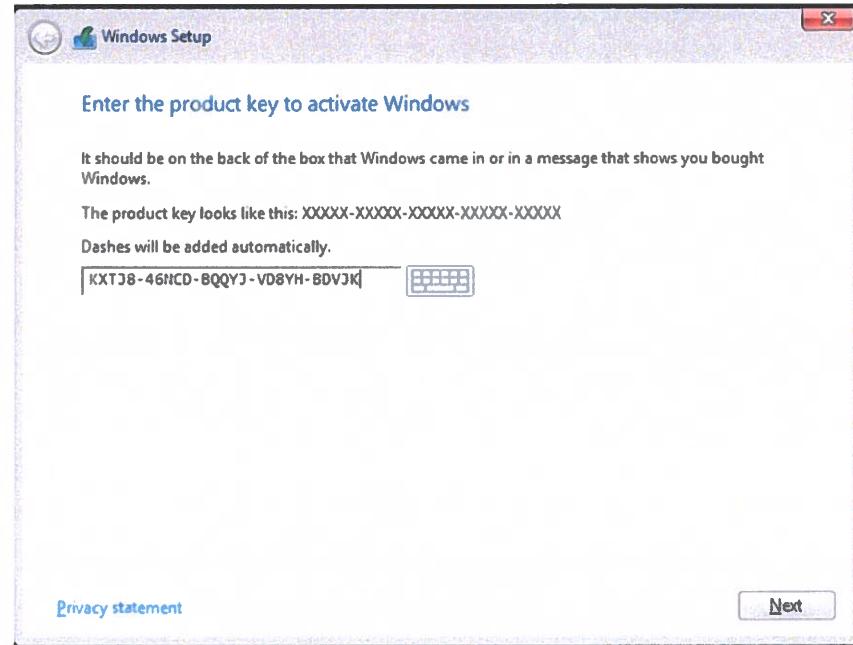


Figura IV- Instalação Sistema Operativo II

Fonte: (do autor)

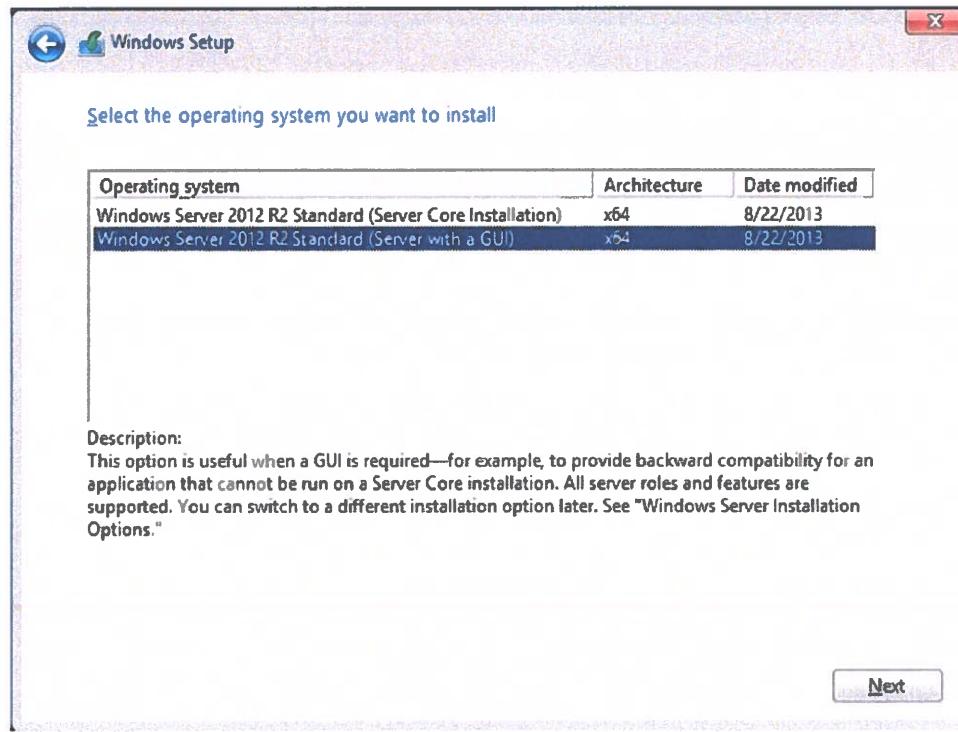


Figura V - Instalação Sistema Operativo III

Fonte: (do autor)

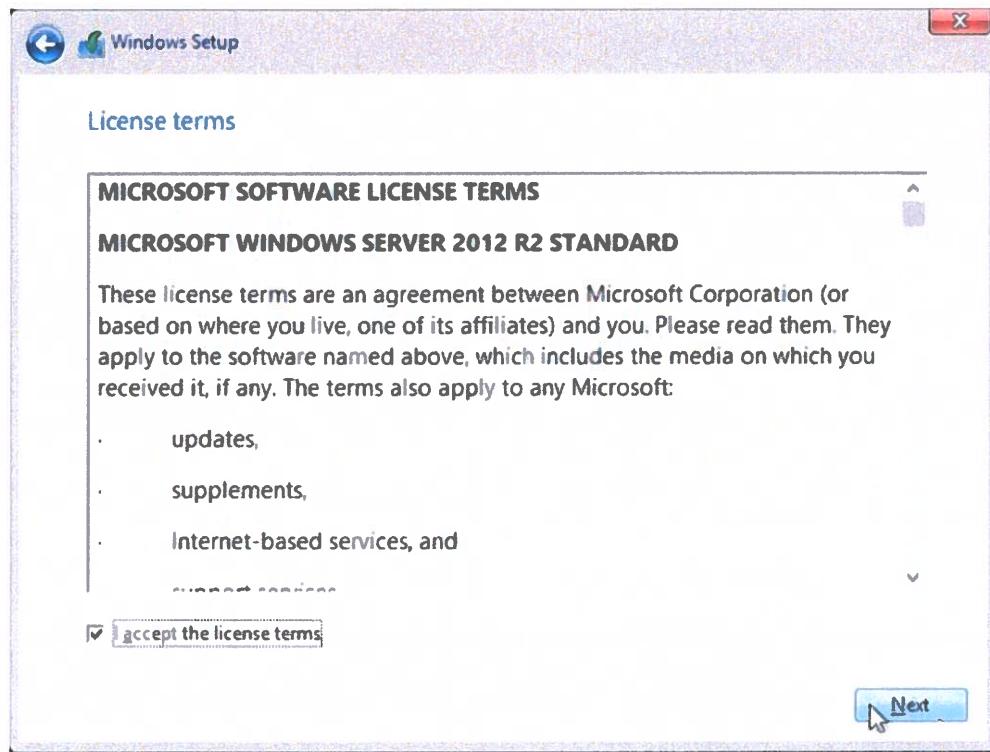


Figura VI - Instalação Sistema Operativo IV

Fonte: (do autor)

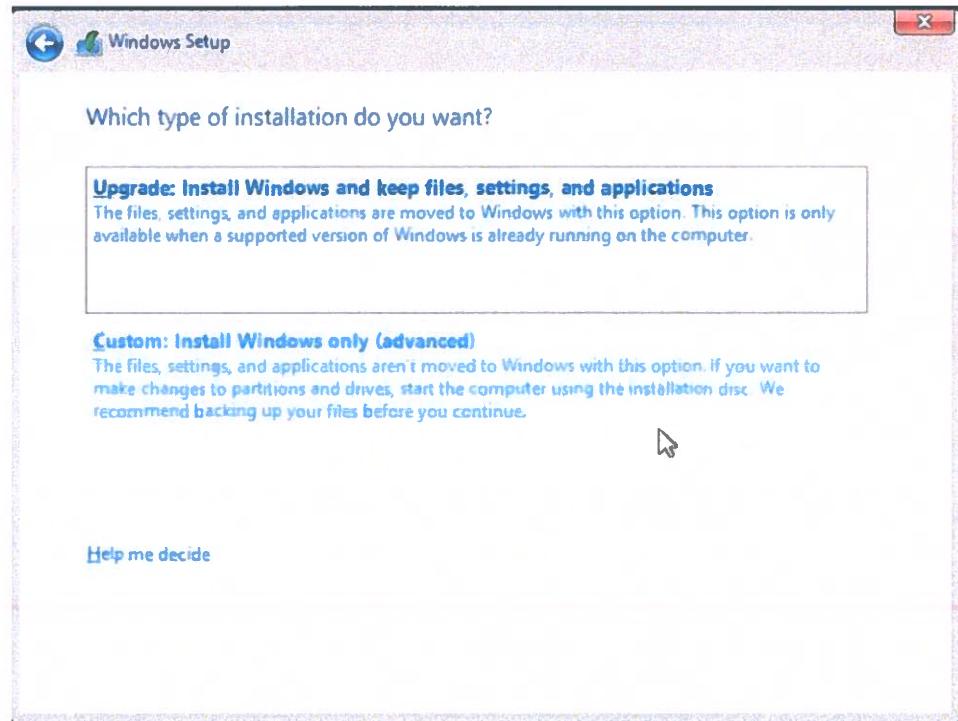


Figura VII - Instalação Sistema Operativo V

Fonte: (do autor)

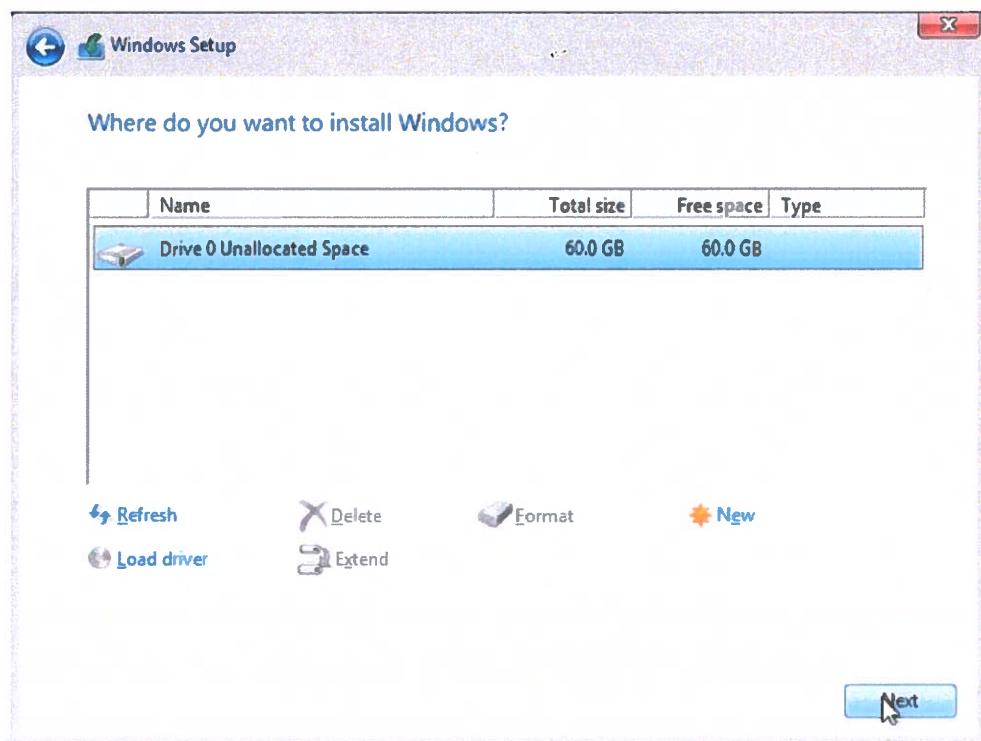


Figura VIII - Instalação Sistema Operativo VI

Fonte: (do autor)

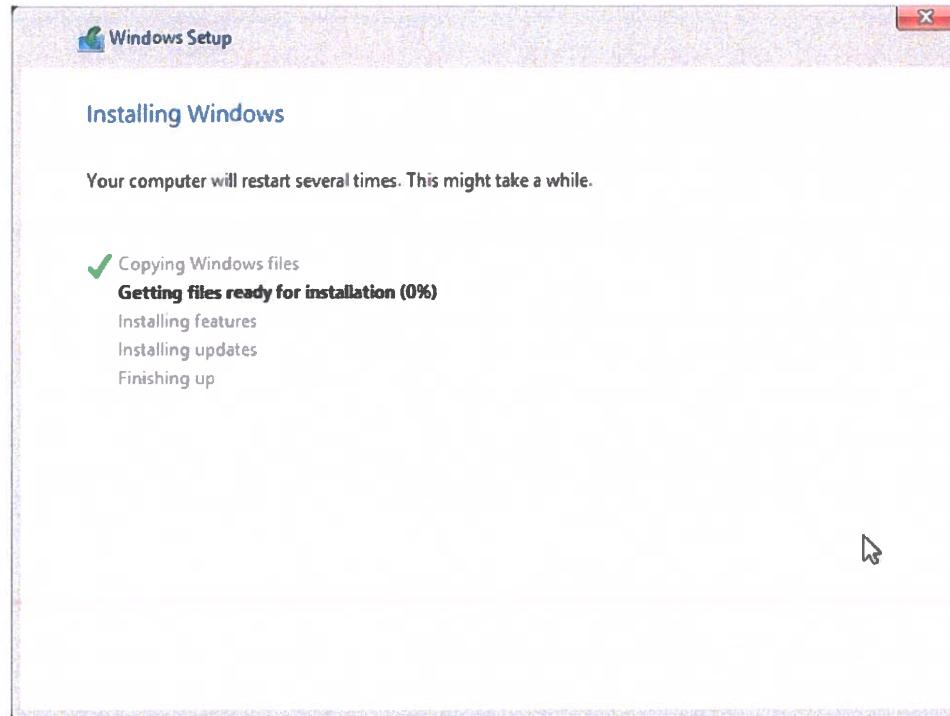


Figura IX - Instalação Sistema Operativo VII

Fonte: (do autor)

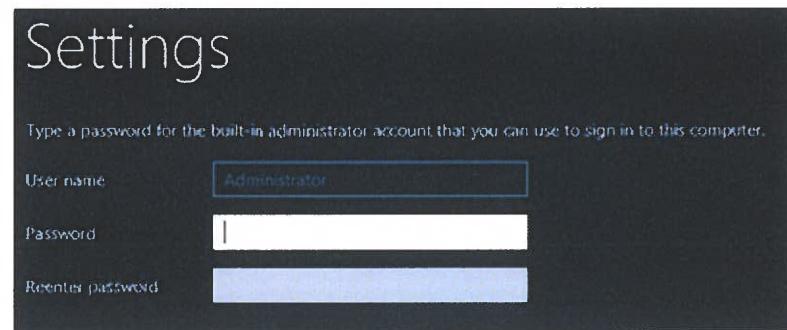


Figura X - Instalação Sistema Operativo VIII

Fonte: (do autor)

Como as imagens indicam o processo consiste na seleção da língua do sistema operativo, do formato da hora e da moeda e da língua do teclado, depois insere-se a chave do sistema operativo, escolhe-se o sistema com interface gráfica (GUI), aceita-se os termos da licença, seleciona-se a opção para instalar apenas o Windows, aguarda-se que os ficheiros sejam instalados e insere-se a palavra passe para a conta *Administrator*.

Na máquina virtual DC, após a instalação do sistema operativo, começa-se pela configuração da interface de rede. Na imagem está ilustrado as configurações da mesma.

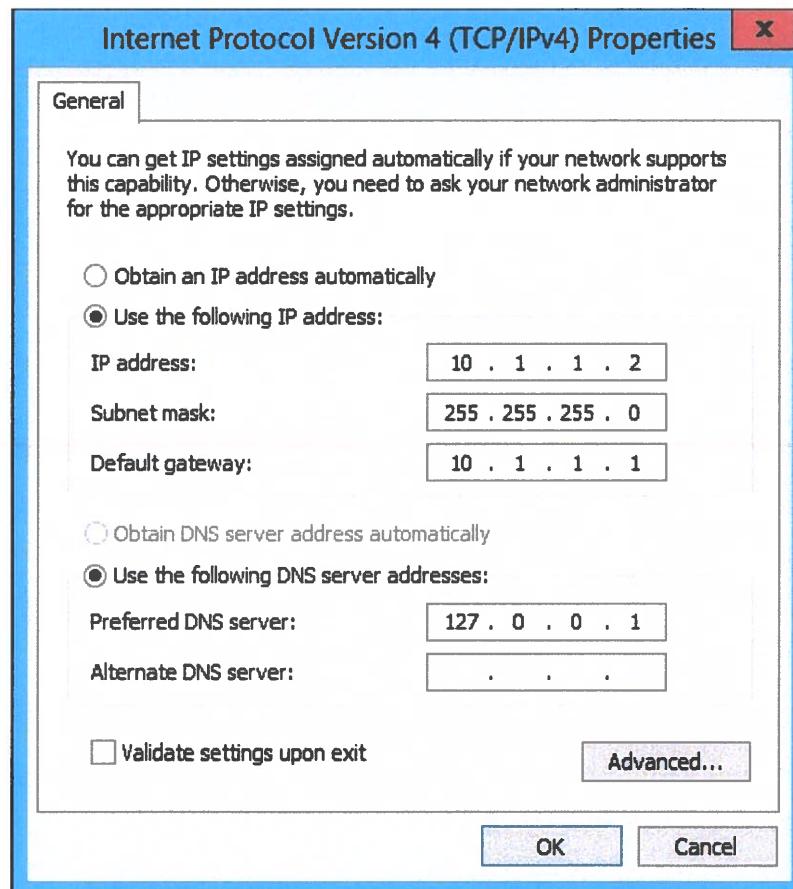


Figura XI - Configuração da interface de rede da Máquina Virtual DC

Fonte: (do autor)

Ao usar o IP 127.0.0.1 no servidor de DNS esta a indicar que o servidor de DNS é ele próprio.

Após a configuração da interface de rede, muda-se o nome da máquina.

Para tal é necessário ir ao Server Manager e selecionar o *Computer Name* e depois *Change*.



Figura XII - Nome da Máquina Virtual e nome do Domínio

Fonte: (do autor)

Depois da mudança do nome é necessário reiniciar a máquina virtual.

Agora vai-se instalar os serviços associados ao Active Directory via PowerShell, é necessário fazer “Run as Administrator”(Correr como Administrador). Escreve-se o comando `Add-WindowsFeature AD-Domain-Services -IncludeManagementTools` no PowerShell aguarda-se a instalação e escreve-se o comando `Install-ADDSForest -DomainName "a053.istec"` para ser criado um domínio com o nome de a053.istec numa nova floresta e promove a máquina DC a controlador de domínio. Será pedido para introduzir uma palavra chave associada à conta de recuperação dos serviços de Active Directory, em caso de falha, tem de se voltar a introduzir a palavra chave, depois insere-se a letra A para aceitar-se tudo e começa o processo de configuração, que inclui a configuração de um servidor DNS. Ao terminar a configuração reinicia-se a máquina virtual e para entrar com a conta do domínio o user tem de ter a seguinte sintaxe `nomedodomínio\contadedomínio`, neste caso fica `a053\Administrator`.

Após a entrada com a conta de domínio, vai se adicionar o DNS *suffix* pois os controladores de domínio podem ter problemas em reconhecer o tipo de rede, a Windows Firewall aparece *Public* em vez de *Domain* e ao adicionar o DNS *suffix*, que será o nome do domínio a053.istec, irá resolver esse problema. Para adicionar, vai-se as definições da interface de rede, seleciona-se o IPv4, depois seleciona-se as opções avançadas e por fim seleciona-se o DNS.



Figura XIII - Configuração do Sufixo do DNS

Fonte: (do autor)

Ao adicionar o sufixo, pode-se constatar na imagem a que o *Windows Firewall* de facto está como *Domain*.

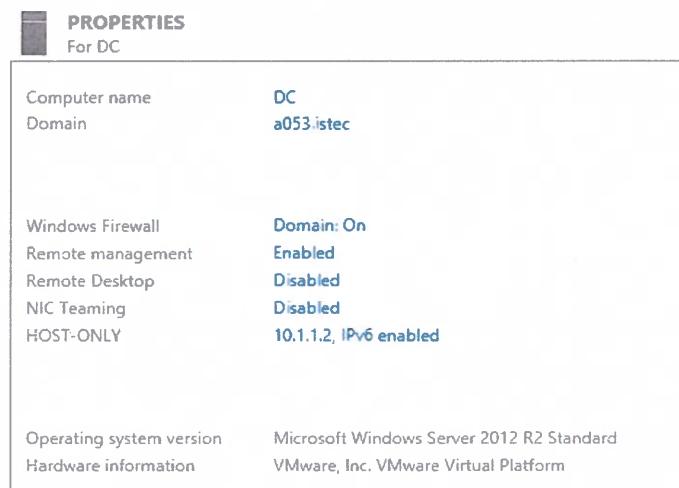


Figura XIV - Windows Firewall Domain: On

Fonte: (do autor)

Agora vem o processo de entrar com as restantes máquinas no domínio, para tal, primeiro tem que se configurar as interfaces de rede *Host-Only* para a mesma gama de IP que o controlador de domínio.

O processo é semelhante em todas as máquinas, o que difere é a atribuição dos IPs, como tal as seguintes imagens demonstram os IPs da máquina *ROUTING*, *SQL* e *CLIENT* pela respetiva ordem.

Use the following IP address:

IP address:

Subnet mask:

Default gateway:
.

Obtain DNS server address automatically

Use the following DNS server addresses:

Preferred DNS server:

Alternate DNS server:
.

Figura XV - Configuração da interface de rede da Máquina Virtual *ROUTING*

Fonte: (do autor)

Use the following IP address:

IP address:

Subnet mask:

Default gateway:

Obtain DNS server address automatically

Use the following DNS server addresses:

Preferred DNS server:

Alternate DNS server:
.

Figura XVI - Configuração da interface de rede da Máquina Virtual *SQL*

Fonte: (do autor)

<input checked="" type="radio"/> Use the following IP address:	
IP address:	10 . 1 . 1 . 5
Subnet mask:	255 . 255 . 255 . 0
Default gateway:	10 . 1 . 1 . 1
<input type="radio"/> Obtain DNS server address automatically	
<input checked="" type="radio"/> Use the following DNS server addresses:	
Preferred DNS server:	10 . 1 . 1 . 2
Alternate DNS server:	. . .

Figura XVII - Configuração da interface de rede da Máquina Virtual CLIENT

Fonte: (do autor)

No servidor DNS coloca-se o IP do controlador de domínio, pois ele também faz o papel de servidor DNS, que é responsável pela resolução de nomes. Na *gateway*, colocamos o IP da máquina *ROUTING*, pois será a responsável pela distribuição do acesso a internet, a partir da placa NAT que já foi referida anteriormente.

Concluída a configuração das interfaces de rede, muda-se o nome e reinicia-se as máquinas virtuais.

Para mudar o nome como foi visto anteriormente basta ir ao *Server Manager*, selecionar o *Computer Name*, depois *Change* e basta mudar o nome da máquina.

É o mesmo processo para a máquina SQL, na *CLIENT* em vez de ir ao *Server Manager* tem de ser ir ao *System*.

Quando o processo de mudar o nome e reiniciar as máquinas estiver terminado, pode-se juntar as máquinas ao domínio. Para tal, vai-se ao mesmo sítio onde se muda o nome e muda-se de *WORKGROUP* para *DOMAIN*, no *DOMAIN* insere-se o nome do domínio.

You can change the name and the membership of this computer. Changes might affect access to network resources.

Computer name:

Full computer name:
Router.a053.istec

Member of

Domain:

Workgroup:

More...

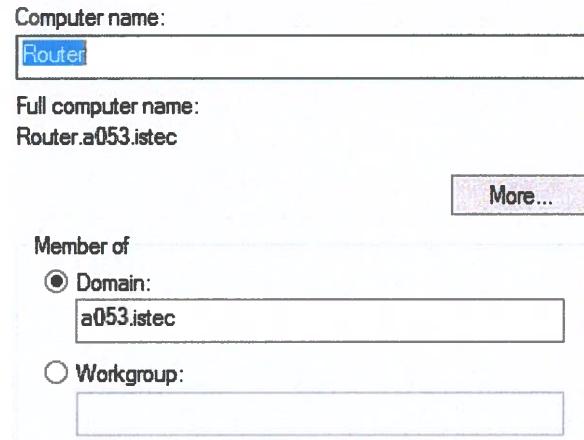


Figura XVIII - Associar Máquina Virtual ao Domínio

Fonte: (do autor)

Irá pedir as credenciais de uma conta que tenha permissão para entrar no domínio.



Figura XIX - Credenciais do Administrador do Domínio

Fonte: (do autor)

Após este processo basta reiniciar a máquina e entrar com uma conta de domínio.

O próximo processo é a configuração da função de *Routing and Remote Access* no servidor *ROUTING*. Para o fazer vai-se ao Server Manager e seleciona-se *Manage* e depois *Add Roles and Features*.

Na página *Before You Begin*, carrega em *Next*, selecionar a opção *Role-based or feature-based installation* e clicar em *Next*, na página *Select destination server*, selecionar o servidor

ROUTING.a053.istec (é onde se pretende instalar o serviço) e clicar em *Next*, na lista de *Server Roles*, selecione *Remote Access* e clicar em *Next*.

Na página *Features* clicar em *Next* e na *Remote Access* também.

Na página *Select role services* ativa-se a caixa de seleção *Routing*, depois clicar em *Add Features* na janela de *pop-up* que surge.

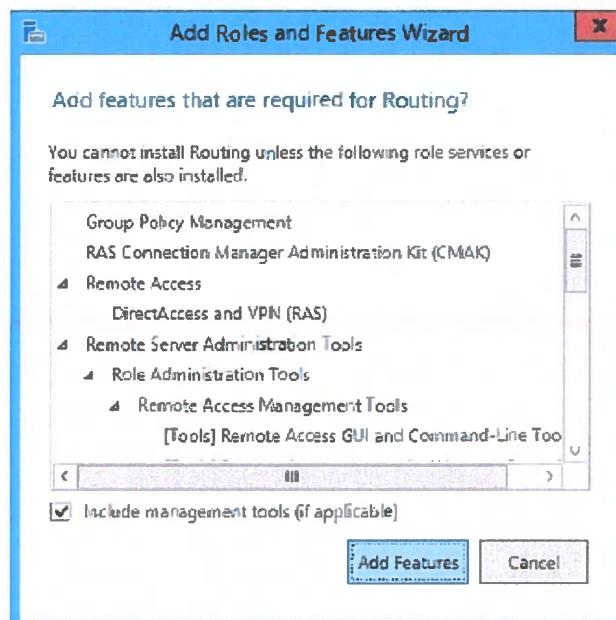


Figura XX - Adicionar o Role de Routing

Fonte: (do autor)

Clicar em *Next*, clicar em *Next* também na página *Web Server Role* (IIS), na página *Role Services* deixar os valores padrão e clicar em *Next*, finalmente, na página *Confirmation* clicar em *Install*.

Aguardar que a *feature* seja instalada. Não será preciso reiniciar o servidor sendo possível avançar para o próximo passo da configuração.

Quando terminar a instalação, pode-se começar com a configuração do serviço de *Routing and Remote Access*. Abre-se a janela *Server Manager* e acede-se a *Tools* e escolhe-se a opção “*Routing and Remote Access*”.

Quando surgir a consola de “*Routing and Remote Access*”, pode verificar-se que no nome do servidor, aparece uma seta vermelha a apontar para baixo.

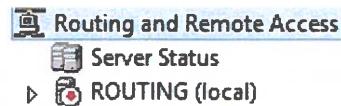


Figura XXI - Estado do Serviço Routing

Fonte: (do autor)

Clicar com o botão direito do rato no nome do servidor e escolher a opção *Configure and Enable Routing and Remote Access*.

O assistente *Routing and Remote Access Server Setup Wizard* aparecerá, clicar em *Next*. No ecrã *Configuration* selecione *Network Address Translation (NAT)*.

Escolhe a interface de rede que está ligado à rede externa ou à *Internet*. Neste caso é a interface de rede NAT, clicar em *NAT* e depois em *Next*, clicar em *Finish* e aguardar que o processo termine e verificar a ligação a *Internet*.

Próximo processo é a instalação do SQL na máquina virtual SQL, para tal, tem de se fazer download dos ficheiros. Foi usado o Microsoft SQL Express 2014 neste laboratório.

No *SQL Server Installation Center* seleciona-se a opção *New SQL Server stand-alone installation or add features to an existing installation*.

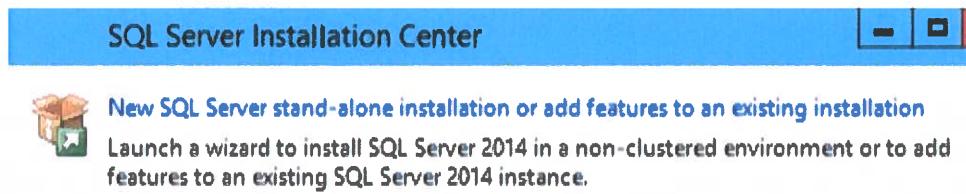


Figura XXII - Instalação do SQL I

Fonte: (do autor)

Escolhe-se uma nova instalação do SQL.

Perform a new installation of SQL Server 2014

Select this option if you want to install a new instance of SQL Server or want to install shared components such as SQL Server Management Studio or Integration Services.

Figura XXIII - Instalação do SQL II

Fonte: (do autor)

Aceita-se os Termos e Condições, seleciona-se as funções a serem instaladas e o diretório onde a instância utilizada para este projeto vai ser guardada. Verifica-se os requerimentos de espaço em disco para que a instalação seja bem-sucedida e segue-se para a próxima página do configurador.

A página seguinte apresenta campos onde permite escolher o nome da instância que foi guardada anteriormente. Foi escolhido o valor por defeito, neste caso é o nome *SQLEXPRESS* como nome de instância.

Na janela de *Service Accounts* a Microsoft recomenda escolher uma conta por serviço, neste caso foi escolhido o *a053\Administrator*, pois para o laboratório não tem influência, embora num ambiente de produção, seria uma falha de segurança e considerado uma má prática.

No tipo de autenticação foi escolhido o Windows *authentication mode*, que faz com que utilizadores e contas de grupo específicas do Windows consigam aceder ao SQL. Também foi escolhido um SQL Server *administrator*, neste caso foi o *a053\Administrator*, mais uma vez, se fosse num ambiente de produção isto seria uma falha de segurança e uma má prática, mas no caso do laboratório não irá afetar.

Após a configuração e instalação do serviço SQL, foi instalado o Microsoft SQL Server Management Studio, que permite fazer a gestão e administração do serviço SQL.

Após a instalação, digita-se as credenciais anteriormente referidas e seleciona-se o tipo de autenticação.

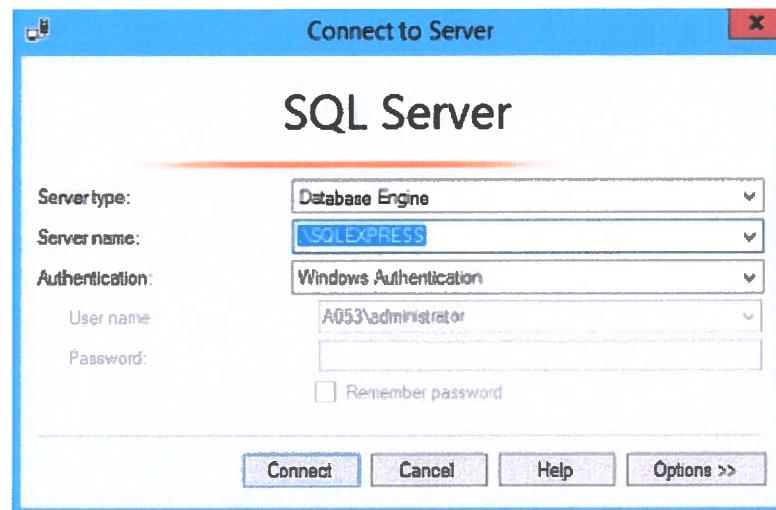


Figura XXIV - Ligar ao servidor SQL

Fonte: (do autor)

Depois feita a conexão, faz-se uma nova *query* para criar a base de dados com o seguinte código.

```
create database istec
on primary (
    NAME='istec_mdf',
    FILENAME ='C:\Program Files\Microsoft SQL Server\MSSQL12.SQLEXPRESS\MSSQL\DATA\istec.mdf',
    SIZE=5MB,
    MAXSIZE=UNLIMITED,
    FILEGROWTH=1MB
)
LOG ON(
    NAME='istec_ldf',
    FILENAME ='C:\Program Files\Microsoft SQL Server\MSSQL12.SQLEXPRESS\MSSQL\DATA\istec.ldf',
    SIZE=5MB,
    MAXSIZE=UNLIMITED,
    FILEGROWTH=1MB
)
```

Figura XXV - Query da criação da base de dados

Fonte: (do autor)

O excerto de código da imagem anterior, faz com que seja criada uma base de dados com o nome istec e um ficheiro de log com o mesmo nome.

Seguindo a criação da base de dados, vem a criação das tabelas.

```
create table empregados(
    idemp int primary key,
    nome varchar(50) not null,
    funcao varchar(15) not null check (funcao in('Técnico', 'Entregas')),
    telemovel bigint not null check (telemovel between 900000000 and 999999999),
    foto varbinary(max),
    fotopath varchar(250),
    documento varchar(250),
);
```

Figura XXVI - Query para criar tabela empregados

Fonte: (do autor)

A *query* anterior, demonstra a criação da tabela empregados, com o campo *idemp*, nome, função, telemóvel, foto, *fotopath* e documento. Alguns dos campos tem restrições. O campo funcao é de preenchimento obrigatório e só aceita Técnico ou Entregas, o telemovel é semelhante, também de preenchimento obrigatório, mas o número tem que ser contido entre 900000000 e 999999999. O campo nome também é de preenchimento obrigatório.

```
create table pedidos(
    idpedido int primary key identity,
    nomecliente varchar(max) not null,
    idemp int foreign key references empregados(idemp)
        on delete cascade on update cascade,
    custo decimal(18,2) not null,
    morada varchar(max) not null,
    relatorio varchar(max) not null,
);
```

Figura XXVII - Query para criar tabela pedidos

Fonte: (do autor)

Nesta *query*, é criada uma tabela com os campos *idpedido*, *nomecliente*, *idemp*, *custo*, *morada*, *relatório*. Todos os campos são de preenchimento obrigatório, o *idemp* é uma chave estrangeira e o *cascade* permite apagar ou atualizar, caso seja apagado ou atualizado o *idemp* da tabela *empregados*. O custo, permite 18 dígitos e vai até duas casas decimais.

Tendo acabado de criar a estrutura da base de dados, pode-se começar a elaborar a aplicação.

Na máquina *CLIENT* foi instalado o Visual Studio 2017, que vai servir para a criação de uma aplicação WPF com ligação á base de dados istec.

Para criar um projeto basta ir a *File > New > Project*, escolher WPF e atribuir um nome ao projeto, neste caso chama-se *FinalProject*.

A aplicação para poder se conectar á base de dados, é necessário adicionar o ADO.NET *Entity Data Model*. As próximas fotos ilustram o processo da adição do mesmo.

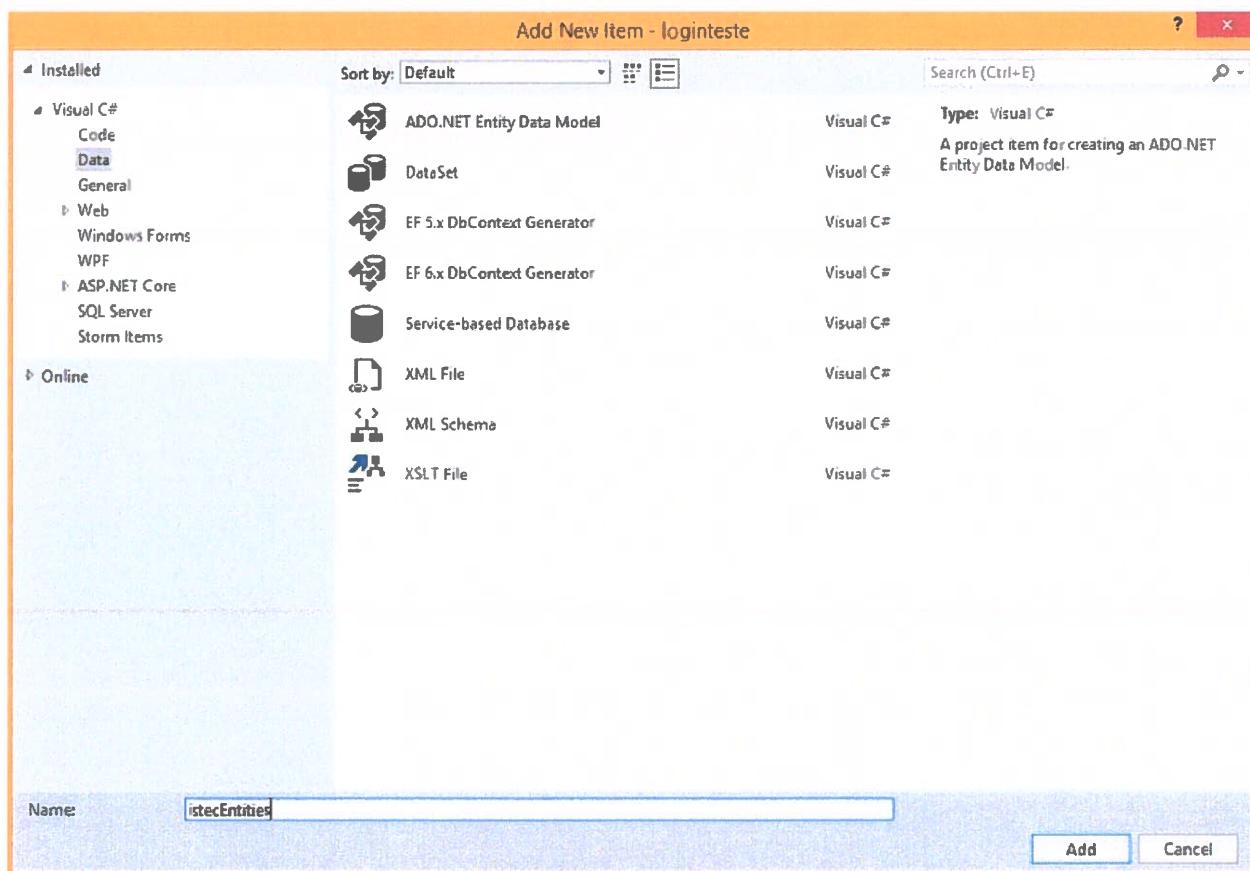


Figura XXVIII - Adicionar Entity Data Model I

Fonte: (do autor)

What should the model contain?

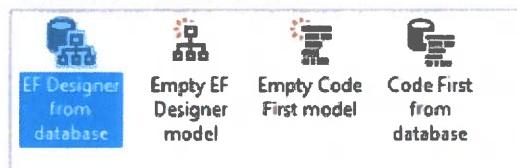


Figura XXIX - Adicionar Entity Data Model II

Fonte: (do autor)

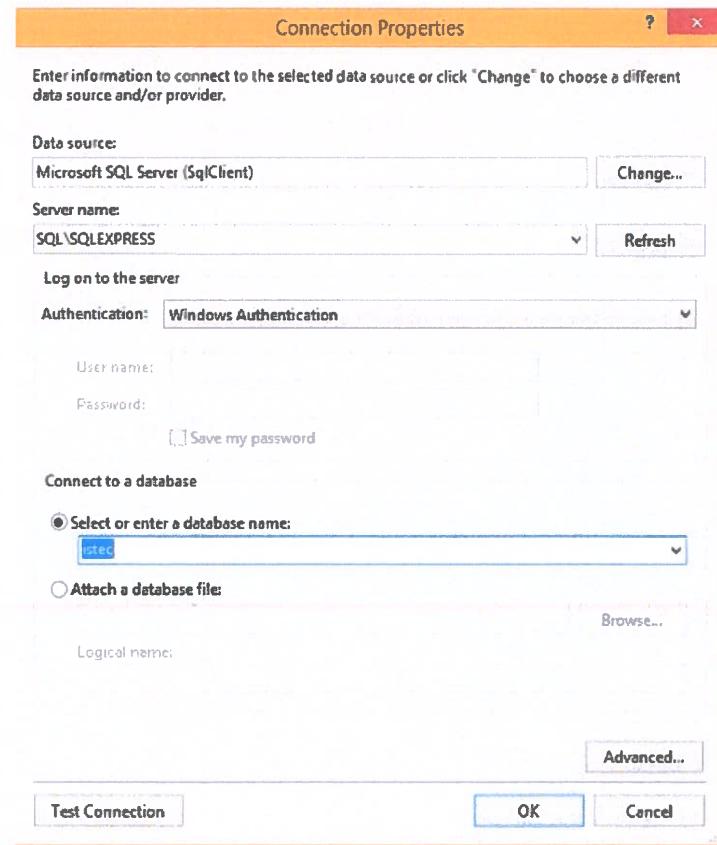


Figura XXX - Adicionar Entity Data Model III

Fonte: (do autor)

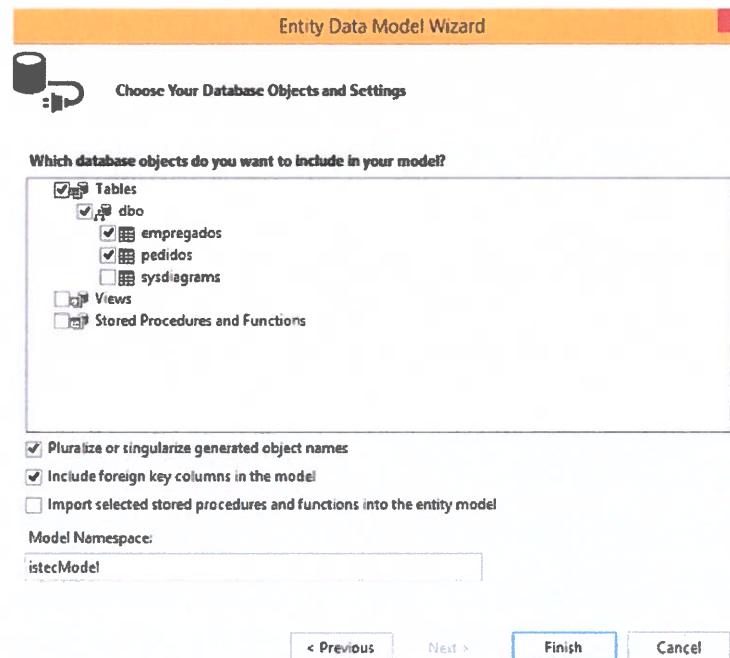


Figura XXXI - Adicionar Entity Data Model IV

Fonte: (do autor)

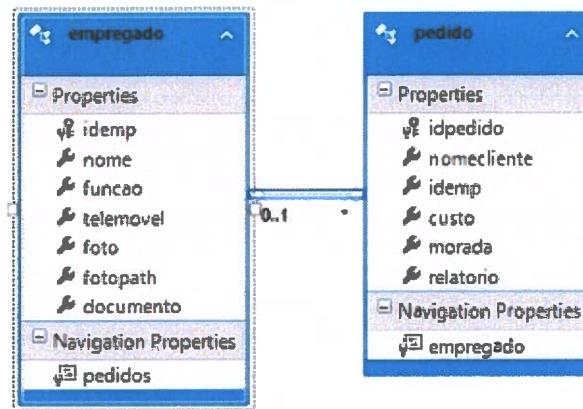


Figura XXXII - Adicionar Entity Data Model V

Fonte: (do autor)

Em suma, vai criar um modelo com as tabelas selecionadas, que permite trabalhar com a base de dados, é também criada uma *connection string* no ficheiro *App.config*.

```
<connectionStrings>
  <add name="istecEntities" connectionString="metadata=res://*/istecEntities.csdl|res://*/istecEntities.ssdl|res://*/istecEntities.msl;provider=System.Data.SqlClient;
  provider connection string="data source=SQL\SQLEXPRESS;initial catalog=istec;integrated security=True;multipleactiveresultsets=True;
  application name=EntityFramework"; providerName="System.Data.EntityClient" />
</connectionStrings>
```

Figura XXXIII - Connection String

Fonte: (do autor)

A primeira janela da aplicação é a do *Login*, cujo aspeto é o seguinte.

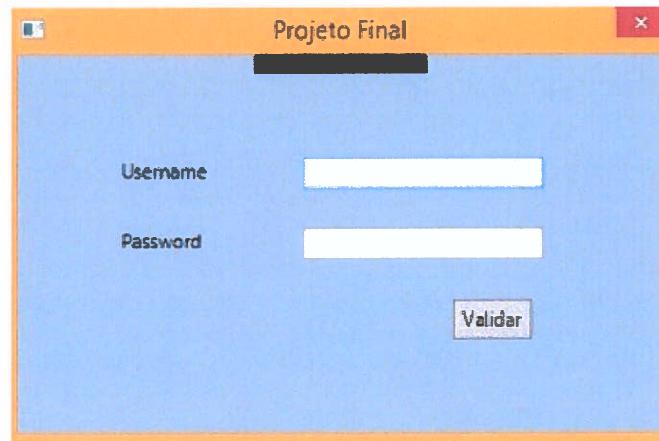


Figura XXXIV - Janela Login

Fonte: (do autor)

O código que permite a autenticação é o seguinte.

```
String usr;
String pw;
String nmdominio = "a053.istec";
MainWindow wind = new MainWindow();

References
public Login()
{
    InitializeComponent();
}

reference
private void Button_Click(object sender, RoutedEventArgs e)
{
    usr = txtlog.Text;
    pw = boxpw.Password.ToString();
    autentica ad = new autentica();

    if (ad.AuthenticateUser(nmdominio, usr, pw))
    {
        this.Close();
        MessageBox.Show("Login Valido");
        wind.ShowDialog();
        //DialogResult = true;
    }
    else
        MessageBox.Show("Login Errado, tente de novo");
    boxpw.Clear();
}
```

Figura XXXV - Botão Validar

Fonte: (do autor)

Tem quatro variáveis globais, o *usr* e a *pw* são obtidos quando se insere os dados, o *nmdominio* tem o valor do nome do domínio e o *wind* é a *MainWindow* que irá ser acedida se o *login* tiver sucesso. A classe *autentica* é a que contém o código do *login*.

```

class autentica
{
    [reference]
    public bool AuthenticateUser(string domainName, string userName, string password)
    {
        bool ret = false;
        try
        {
            DirectoryEntry de = new DirectoryEntry("LDAP://" + domainName, userName, password);
            DirectorySearcher dsearch = new DirectorySearcher(de);
            SearchResult results = null;
            results = dsearch.FindOne();
            ret = true;
        }
        catch
        {
            ret = false;
        }
        return ret;
    }
}

```

Figura XXXVI - Classe autêntica

Fonte: (do autor)

A classe autentica, vai procurar o utilizador e a password no domínio com o auxílio do protocolo *LDAP*, que serve para guardar informação sobre utilizadores, grupos, aplicações de forma centralizada e hierárquica e também permite autenticação de utilizadores. Se retornar falso aparece uma mensagem a dizer “*Login Errado, tente de novo*”, caso retorne verdadeiro, aparece uma mensagem a dizer “*Login Valido*” e é redirecionado para a janela *MainWindow*.

A janela *MainWindow* contem três páginas. A primeira é a seguinte.

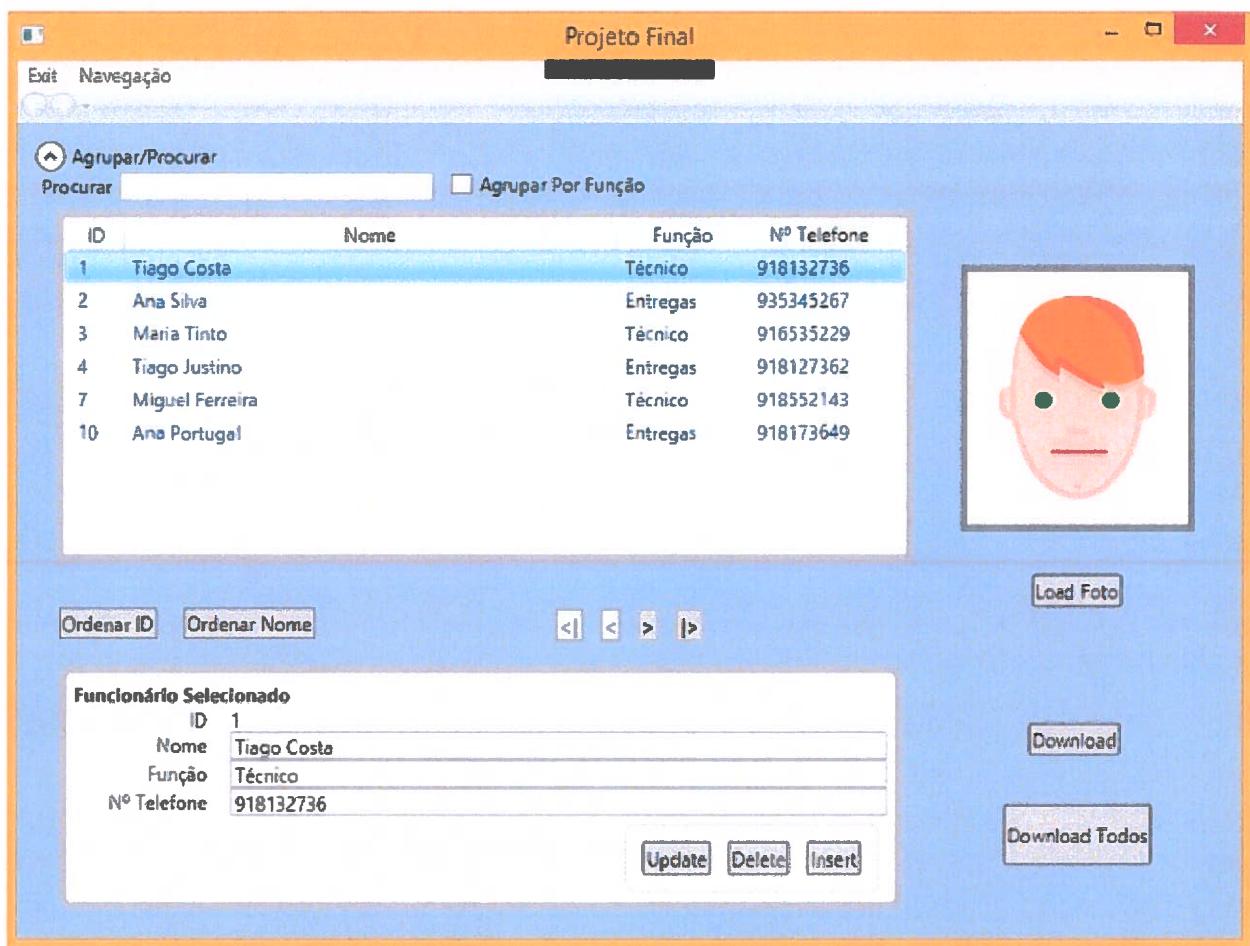


Figura XXXVII - Page1

Fonte: (do autor)

Na Page1 tem uma *Listview* em que mostra os funcionários da empresa, para tal foi criada uma classe chamada *ViewModel*, onde contém o código para fazer a ligação á base de dados. A classe é instanciada no *xaml* com o nome *model* e é criado o *binding* da função *ListEmp* na *ListView*. Foi também instanciada a classe conversor com o nome *xpto*, que contem o código que vai buscar a imagem a partir do caminho da mesma.

```
<Window.Resources>
<local:ViewModel x:Key="model"></local:ViewModel>
</Window.Resources>
```

Figura XXXVIII - XAML MainWindow ViewModel

Fonte: (do autor)

```

<Page.Resources>
    <local:conversor x:Key="xpto"></local:conversor>
</Page.Resources>

```

Figura XXXIX - XAML Conversor Page1

Fonte: (do autor)

```

<ListView Name="lst" Grid.Row="3" BorderBrush="Silver" BorderThickness="5" ItemsSource="{Binding ListEmp}"

```

Figura XL - XAML Page1 ListView ListEmp

Fonte: (do autor)

```

listEmp = new ObservableCollection<empregado>(EmpModel.empregados);
ViewEmp = CollectionViewSource.GetDefaultView(_listEmp);
ListEmp.CollectionChanged += ListEmp_CollectionChanged;
ViewEmp.CurrentChanged += ViewEmp_CurrentChanged;
EmpCorrente = (empregado)ViewEmp.CurrentItem;

```

Figura XLI - ListEmp ViewModel

Fonte: (do autor)

Com excertos de código é possível ver os empregados na lista, sabendo o que está selecionado, permitindo mostrar a imagem associada ao mesmo, apagar ou atualizar o registo do selecionado.

```

<TextBlock Text="ID" Grid.Row="1" Grid.Column="0" HorizontalAlignment="Right" Margin="0,0,15,0" />
<TextBlock Name="txtid" Text="{Binding EmpCorrente.idemp, Mode=TwoWay}" Grid.Row="1" Grid.Column="1"/>

<TextBlock Text="Nome" Grid.Row="2" Grid.Column="0" HorizontalAlignment="Right" Margin="0,0,15,0" />
<TextBox Name="txtnome" Text="{Binding EmpCorrente.nome, Mode=TwoWay}" Grid.Column="1" Grid.Row="2"/>

<TextBlock Text="Função" Margin="0,0,15,0" Grid.Row="3" Grid.Column="0" HorizontalAlignment="Right" />
<TextBox Name="txtfunc" Text="{Binding EmpCorrente.funcao, Mode=TwoWay}" Grid.Column="1" Grid.Row="3"/>

<TextBlock Text="Nr Telefone" Margin="0,0,15,0" Grid.Row="4" Grid.Column="0" HorizontalAlignment="Right" />
<TextBox Name="txttele" Text="{Binding EmpCorrente.teleovel, Mode=TwoWay}" Grid.Column="1" Grid.Row="4"/>

```

Figura XLII - TextBlocks com o Empregado Corrente

Fonte: (do autor)

```

<Image Name="img" VerticalAlignment="Center" Source="{Binding EmpCorrente.fotopath, Converter={StaticResource xpto}}" />
```

Figura XLIII - Imagem do Empregado Corrente

Fonte: (do autor)

No topo da *Page1* tem uma *TextBox* que permite procurar o nome do empregado.

```

<TextBox Name="txtproc" Width="200" HorizontalAlignment="Left" Text="{Binding SearchText, UpdateSourceTrigger=PropertyChanged}"></TextBox>
```

Figura XLIV - TextBox Procura Nome

Fonte: (do autor)

```

private string _searchText;
1 reference
public string SearchText
{
    get { return _searchText; }
    set
    {
        _searchText = value;
        ViewEmp.Filter = FilterData;
        OnPropertyChanged("SearchText");
    }
}

1 reference
private bool FilterData(object item)
{
    var value = (empregado)item;
    if (value == null || value.nome == null)
        return false;

    return value.nome.Contains(SearchText);
}

```

Figura XLV - Código para filtrar nome ViewModel

Fonte: (do autor)

Atualiza o *ViewEmp* com o valor da procura se existir, dessa maneira a *ListView* também é atualizada.

Também no topo da *Page1*, tem uma *Checkbox* que permite agrupar os empregados por função.

```
<CheckBox Content="Agrupar Por Função" Margin="10,0,0,0" IsChecked="{Binding GroupByClube}" />
```

Figura XLVI - CheckBox Filtrar por Função

Fonte: (do autor)

```

private bool _groupByClube;
1 reference
public bool GroupByClube
{
    get { return _groupByClube; }
    set
    {
        _groupByClube = value;
        GroupData();
        OnPropertyChanged("GroupByClube");
    }
}
1 reference
private void GroupData()
{
    using (ViewEmp.DeferRefresh())
    {
        ViewEmp.GroupDescriptions.Clear();
        if (_groupByClube) ViewEmp.GroupDescriptions.Add(new PropertyGroupDescription("funcao"));
    }
}

```

Figura XLVII - Código para agrupar por Função

Fonte: (do autor)

Muda o *ViewEmp* para mostrar os empregados agrupados por função.

Tem dois botões para ordenar pelo *Id* ou pelo nome.

```
<Button Content="Ordenar ID" Margin="10,0,0,0" HorizontalAlignment="Left" Command="local:Comandos.ordenarid" VerticalAl  
<Button HorizontalAlignment="Left" Content="Ordenar Nome" VerticalAlignment="Top" Command="local:Comandos.ordenarnome"
```

Figura XLVIII - Botão ordenar *Id* e ordenar nome

Fonte: (do autor)

O código da função é bastante semelhante, só muda a variável, num é *idemp* noutro é nome, também muda o nome da função.

```
private bool _directionAscending = true;  
referencia  
public void ordenarid()  
{  
  
    if (_directionAscending == true)  
    {  
        _directionAscending = false;  
    }  
    else  
    {  
        _directionAscending = true;  
    }  
  
    ListSortDirection direccao = (_directionAscending) ? ListSortDirection.Ascending : ListSortDirection.Descending;  
    using (ViewEmp.DeferRefresh())  
    {  
        ViewEmp.SortDescriptions.Clear();  
        ViewEmp.SortDescriptions.Add(new SortDescription("idemp", direccao));  
    }  
}
```

Figura XLIX - Função ordenar *Id*

Fonte: (do autor)

Este excerto permite ver os empregados organizados por *Id* de forma ascendente ou descendente. A função ordenarnome faz o mesmo, mas com os nomes dos empregados.

O botão *Load Foto* permite atualizar a foto do empregado corrente.

```
<Button Content="Load Foto" HorizontalAlignment="Right" Command="local:Comandos.loadimage" VerticalAlignment="Top"></Button>
```

Figura L - Botão Load Foto

Fonte: (do autor)

Em todos os casos que haja a função *Command* tem de se referenciar a função na classe Comandos para poder ir buscar a função ao *ViewModel*.

```

public static class Comandos
{
    public static RoutedUICommand sair = new RoutedUICommand("Sair", "sair", typeof(Comandos));
    public static RoutedUICommand navegarpag1 = new RoutedUICommand("Navegarpag1", "navegarpag1", typeof(Comandos));
    public static RoutedUICommand navegarpag2 = new RoutedUICommand("Navegarpag2", "navegarpag2", typeof(Comandos));
    public static RoutedUICommand navegarpag3 = new RoutedUICommand("Navegarpag3", "navegarpag3", typeof(Comandos));
    public static RoutedUICommand gofirst = new RoutedUICommand("Go First", "gofirst", typeof(Comandos));
    public static RoutedUICommand gonext = new RoutedUICommand("Go Next", "gonext", typeof(Comandos));
    public static RoutedUICommand golast = new RoutedUICommand("Go Last", "golast", typeof(Comandos));
    public static RoutedUICommand goprevious = new RoutedUICommand("Go Previous", "goprevious", typeof(Comandos));
    public static RoutedUICommand updateEmp = new RoutedUICommand("updateEmp", "updateEmp", typeof(Comandos));
    public static RoutedUICommand deleteEmp = new RoutedUICommand("deleteEmp", "deleteEmp", typeof(Comandos));
    public static RoutedUICommand inserirEmp = new RoutedUICommand("inserirEmp", "inserirEmp", typeof(Comandos));
    public static RoutedUICommand loadimagem = new RoutedUICommand("loadimagem", "loadimagem", typeof(Comandos));
    public static RoutedUICommand insertimagem = new RoutedUICommand("insertimagem", "insertimagem", typeof(Comandos));
    public static RoutedUICommand deletepedido = new RoutedUICommand("deletepedido", "deletepedido", typeof(Comandos));
    public static RoutedUICommand inserirpedido = new RoutedUICommand("inserirpedido", "inserirpedido", typeof(Comandos));
    public static RoutedUICommand updatepedido = new RoutedUICommand("updatepedido", "updatepedido", typeof(Comandos));
    public static RoutedUICommand guardaPdf = new RoutedUICommand("guardaPdf", "guardaPdf", typeof(Comandos));
    public static RoutedUICommand PdfComTodos = new RoutedUICommand("PdfComTodos", "PdfComTodos", typeof(Comandos));
    public static RoutedUICommand ordenarid = new RoutedUICommand("ordenarid", "ordenarid", typeof(Comandos));
    public static RoutedUICommand ordenarnome = new RoutedUICommand("ordenarnome", "ordenarnome", typeof(Comandos));
}

```

Figura LI - Classe Comandos

Fonte: (do autor)

```

CommandBindings.Add(new CommandBinding(
    Comandos.updateEmp,
    (sender, e) => vm.updateEmp(vm.EmpCorrente),
    (sender, e) => e.CanExecute = true
));

CommandBindings.Add(new CommandBinding(
    Comandos.deleteEmp,
    (sender, e) => vm.deleteEmp(vm.EmpCorrente),
    (sender, e) => e.CanExecute = true
));
CommandBindings.Add(new CommandBinding(
    Comandos.loadimagem,
    (sender, e) => vm.loadimagem(img, vm.EmpCorrente),
    (sender, e) => e.CanExecute = true
));

```

Figura LII - CommandBiding PageI

Fonte: (do autor)

O processo é sempre o mesmo, o que muda são as funções de cada um. A função que permite atualizar a imagem é a seguinte.

```

public void lqadimagem(System.Windows.Controls.Image img, empregado a)
{
    string oldfich = "";
    OpenFileDialog dlg = new OpenFileDialog();
    dlg.InitialDirectory = @"C:\Users\Administrator\Desktop";
    dlg.Filter = "*.*|*.JPEG Files (*.jpeg)|*.jpg|*.PNG Files (*.png)|*.png|*.JPG Files (*.jpg)|*.jpg|*.GIF Files (*.gif)|*.gif";
    if (dlg.ShowDialog() == true)
    {
        try
        {
            string fich = Environment.CurrentDirectory;
            fich = fich.Substring(0, fich.IndexOf("bin")) + "imagens\\" + EmpCorrente.idemp.ToString() + ".png";
            if (EmpCorrente.fotopath != null & EmpCorrente.fotopath == Path.GetFileName(fich))
            {
                oldfich = fich;
                fich = fich.Replace(@".", @"B.");
            }
            EmpCorrente.fotopath = Path.GetFileName(fich);

            File.Copy(dlg.FileName, fich, true);
            onPropertyChanged("EmpCorrente");
            updateEmp(EmpCorrente);
        }
        catch (Exception erro) { MessageBox.Show(erro.Message); }
    }
}

```

Figura LIII - Código para atualizar a imagem do empregado corrente

Fonte: (do autor)

Ele deteta o diretório em que esta, acrescenta a pasta imagens ao caminho da foto e designa o nome da imagem pelo *Id* do empregado, ex: 1.png, associando a imagem com o empregado corrente.

Os botões *Download* e *Download Todos* permitem fazer o *download* do registo dos pedidos de cada empregado ou de todos em formato Pdf.

```

<Border CornerRadius="3" BorderBrush="Gray" BorderThickness="1" HorizontalAlignment="Right" Margin="0,-120,60,120">
    <Button Content="Download" Command="local:Comandos.guardaPdf" HorizontalAlignment="Right" Grid.Row="4" ></Button>
</Border>
<Border CornerRadius="3" BorderBrush="Gray" BorderThickness="1" HorizontalAlignment="Right" Margin="0,-90,40,50">
    <Button Content="Download Todos" Command="local:Comandos.PdfComTodos" HorizontalAlignment="Right" Grid.Row="4" ></Button>
</Border>

```

Figura LIV - Botões Download e Download Todos

Fonte: (do autor)

O código para o botão *Downlaod* é o seguinte.

```

public void guardaPdf(int num)
{
    string caminho = Environment.CurrentDirectory;
    caminho = caminho.Substring(0, caminho.IndexOf("bin")) + "documentos\\" +
num.ToString() + ".pdf";
    String fotopath = (EmpCorrente.fotopath != null) ?
Environment.CurrentDirectory.Substring(0, Environment.CurrentDirectory.IndexOf("bin")) +
"imagens\\" + EmpCorrente.fotopath :
Environment.CurrentDirectory.Substring(0,
Environment.CurrentDirectory.IndexOf("bin")) + "imagens\\nofoto.png";
    Document doc = new Document(PageSize.A4, 15f, 15f, 15f, 15f);
    try
    {
        FileStream fs = new FileStream(caminho, FileMode.Create, FileAccess.Write);
        PdfWriter w = PdfWriter.GetInstance(doc, fs);
        doc.Open();
        Font titulo1 = FontFactory.GetFont("COURIER", 20f);

```

```

        titulo1.SetStyle(Font.BOLD);
        Paragraph p1 = new Paragraph("Registro do Funcionario\n", titulo1);
        p1.Alignment = Element.ALIGN_CENTER;
        doc.Add(p1);
        doc.Add(new Phrase("\n\n"));
        var foto = iTextSharp.text.Image.GetInstance(fotopath);
        foto.Alignment = iTextSharp.text.Image.ALIGN_CENTER;
        foto.ScaleAbsoluteHeight(200);
        foto.ScaleAbsoluteWidth(170);
        doc.Add(foto);
        doc.Add(new Phrase("\n\n\n"));
        Font titulo2 = FontFactory.GetFont("COURIER", 15f);
        Paragraph p2 = new Paragraph("Id: "+EmpCorrente.idemp.ToString() + "    Nome:
" + EmpCorrente.nome, titulo2);
        p2.IndentationLeft = 120;
        p2.SetLeading(5.0f, 4.0f);
        doc.Add(p2);
        doc.Add(new Phrase("\n\n"));
        PdfPTable table = new PdfPTable(6);
        table.DefaultCell.BorderColor = BaseColor.BLACK;
        table.DefaultCell.Border = Rectangle.BOX;
        table.DefaultCell.HorizontalAlignment = Element.ALIGN_LEFT;
        PdfPCell cell = new PdfPCell(new Phrase("Pedidos do Funcionario", titulo1));
        cell.Colspan = 6;
        cell.HorizontalAlignment = Element.ALIGN_CENTER;
        cell.VerticalAlignment = Element.ALIGN_CENTER;
        table.AddCell(cell);
        table.AddCell(new Phrase("Nº Pedido"));
        table.AddCell(new Phrase("IdEmp"));
        table.AddCell(new Phrase("Nome Cliente"));
        table.AddCell(new Phrase("Custo"));
        table.AddCell(new Phrase("Morada"));
        table.AddCell(new Phrase("Relatorio"));
        foreach (pedido n in EmpCorrente_pedidos)
        {
            table.AddCell(new Phrase(n.idpedido.ToString()));
            table.AddCell(new Phrase(n.idemp.ToString()));
            table.AddCell(new Phrase(n.nomecliente));
            table.AddCell(new Phrase(n.custo.ToString()));
            table.AddCell(new Phrase(n.morada));
            table.AddCell(new Phrase(n.relatorio));
        }
        doc.Add(table);
        EmpCorrente_documento = caminho;
        Process.Start(@caminho);
    }
    catch (DocumentException dex)
    {
        throw (dex);
    }
    catch (IOException ioex)
    {
        throw (ioex);
    }
    finally
    {
        doc.Close();
    }
}

```

A função guardaPdf cria um documento Pdf com o nome do Id do empregado que contem o Id, o nome, a foto e uma tabela com os pedidos do empregado corrente.

O código para o botão *Download Todos* é o seguinte.

```
public void PdfComTodos()
{
    int pagenumber = 1;
    Document doc = new Document(PageSize.A4, 15f, 15f, 15f, 15f);
    try
    {
        string caminho = Environment.CurrentDirectory;
        caminho = caminho.Substring(0, caminho.IndexOf("bin")) +
"documentos\\todos.pdf";
        FileStream fs = new FileStream(caminho, FileMode.Create, FileAccess.Write);
        PdfWriter w = PdfWriter.GetInstance(doc, fs);
        doc.Open();
        foreach (empregado emp in ListEmp)
        {

            String fotopath = (emp.fotopath != null) ?
Environment.CurrentDirectory.Substring(0, Environment.CurrentDirectory.IndexOf("bin")) +
"imagens\\" + emp.fotopath :
Environment.CurrentDirectory.Substring(0,
Environment.CurrentDirectory.IndexOf("bin")) + "imagens\\nofoto.png";
            Font titulo1 = FontFactory.GetFont("COURIER", 20f);
            titulo1.SetStyle(Font.BOLD);
            Paragraph p1 = new Paragraph("Registo do Funcionario\n", titulo1);
            p1.Alignment = Element.ALIGN_CENTER;
            doc.Add(p1);
            doc.Add(new Phrase("\n\n"));
            var foto = iTextSharp.text.Image.GetInstance(fotopath);
            foto.Alignment = iTextSharp.text.Image.ALIGN_CENTER;
            foto.ScaleAbsoluteHeight(200);
            foto.ScaleAbsoluteWidth(170);
            doc.Add(foto);
            doc.Add(new Phrase("\n\n\n"));
            Font titulo2 = FontFactory.GetFont("COURIER", 15f);
            Paragraph p2 = new Paragraph("Id: " + emp.idemp.ToString() + "     Nome: " +
emp.nome, titulo2);
            p2.IndentationLeft = 120;
            p2.SetLeading(5.0f, 4.0f);
            doc.Add(p2);
            doc.Add(new Phrase("\n\n"));
            PdfPTable table = new PdfPTable(6);
            table.DefaultCell.BorderColor = BaseColor.BLACK;
            table.DefaultCell.Border = Rectangle.BOX;
            table.DefaultCell.HorizontalAlignment = Element.ALIGN_LEFT;
            PdfPCell cell = new PdfPCell(new Phrase("Pedidos do Funcionario",
titulo1));
            cell.Colspan = 6;
            cell.HorizontalAlignment = Element.ALIGN_CENTER;
            cell.VerticalAlignment = Element.ALIGN_CENTER;
            table.AddCell(cell);
            table.AddCell(new Phrase("Nº Pedido"));
            table.AddCell(new Phrase("IdEmp"));
            table.AddCell(new Phrase("Nome Cliente"));
            table.AddCell(new Phrase("Custo"));
            table.AddCell(new Phrase("Morada"));
            table.AddCell(new Phrase("Relatorio"));
            foreach (pedido n in emp_pedidos)
```

```

    {
        table.AddCell(new Phrase(n.idpedido.ToString()));
        table.AddCell(new Phrase(n.idemp.ToString()));
        table.AddCell(new Phrase(n.nomecliente));
        table.AddCell(new Phrase(n.custo.ToString()));
        table.AddCell(new Phrase(n.morada));
        table.AddCell(new Phrase(n.relatorio));
    }
    doc.Add(table);
    Paragraph footer = new Paragraph(".....");
    footer.SetLeading(0f, 18f);
    doc.Add(footer);
    LineSeparator line = new LineSeparator(1f, 100f, BaseColor.BLACK,
Element.ALIGN_LEFT, 1);
    doc.Add(line);
    doc.Add(new Paragraph(pagename.ToString()));
    doc.NewPage();
    pagename++;
}
Process.Start(@caminho);
}
catch (DocumentException dex)
{
    throw (dex);
}
catch (IOException ioex)
{
    throw (ioex);
}
finally
{
    doc.Close();
}

}

```

A função PdfComTodos cria um documento Pdf com o nome de todos, que contem todos os empregados, os seus *Ids*, nomes, fotos e a tabela com os respetivos pedidos.

Tem quatro botões que permitem navegar entre os empregados, primeiro, último, próximo e anterior.

```

<Button BorderBrush="Silver" BorderThickness="2" Content="&lt;|" Margin="5" Command="local:Comandos.gofirst" FontWeight="Bold" />
<Button BorderBrush="Silver" BorderThickness="2" Content="&lt;" Margin="5" Command="local:Comandos.goprevious" FontWeight="Bold" />
<Button BorderBrush="Silver" BorderThickness="2" Content="&gt;" Margin="5" Command="local:Comandos.gonext" FontWeight="Bold" />
<Button BorderBrush="Silver" BorderThickness="2" Content="|&gt;" Margin="5" Command="local:Comandos.golast" FontWeight="Bold" />

```

Figura LV - Botões para navegar na ListView

Fonte: (do autor)

O código dos botões é o seguinte.

```

public bool canGoFirst()
{
    return ViewEmp.CurrentPosition > 0;
}
public bool canGoLast()
{
    return ViewEmp.CurrentPosition < (ViewEmp.Cast<empregado>()).Count() -
1;
}
public void goFirst()
{
    ViewEmp.MoveCurrentToFirst();
    EmpCorrente = (empregado)ViewEmp.CurrentItem;
}
public void goLast()
{
    ViewEmp.MoveCurrentToLast();
    EmpCorrente = (empregado)ViewEmp.CurrentItem;
}
public void goNext()
{
    ViewEmp.MoveCurrentToNext();
    EmpCorrente = (empregado)ViewEmp.CurrentItem;
}
public void goPrevious()
{
    ViewEmp.MoveCurrentToPrevious();
    EmpCorrente = (empregado)ViewEmp.CurrentItem;
}

```

Por cada ação tomada, atualiza o *ViewEmp* e o *EmpCorrente*, seja o primeiro, o último, o seguinte ou o anterior.

O botão *Update* serve para atualizar os campos do empregado corrente, neste caso, o nome, o telemóvel e a função.

```
<button Content="Update" DockPanel.Dock="Right" Command="local:Comandos.updateEmp" HorizontalAlignment="Right" ></button>
```

Figura LVI - Botão Update

Fonte: (do autor)

O código do botão *Update* é o seguinte.

```

public void updateEmp(empregado a)
{
    try
    {
        var este = EmpModel.empregados.First(x => x.idemp == a.idemp);
        if (este != null)
        {

            este.idemp = a.idemp;
            este.nome = a.nome;
            este.funcao = a.funcao;
            este.telemovel = a.telemovel;
            if (a.fotopath != null) este.fotopath = a.fotopath;
            if (este.nome != "" && este.funcao != "")
            {
                EmpModel.SaveChanges();
                MessageBox.Show("Atualizado com Sucesso");
            }
            else MessageBox.Show("Todos os campos são de preenchimento
obrigatorio");
        }
    }
    catch
    {
        MessageBox.Show("Verificar os Campos");
    }
}

```

Atualiza as informações do empregado corrente/selecionado com as novas informações que recebe das *TextBox* e grava na base de dados.

O botão *Delete* apaga o empregado corrente/selecionado e todos os campos associados a ele.

```
<Button Content="Delete" DockPanel.Dock="Right" Command="local:Comandos.deleteEmp" HorizontalAlignment="Right"></Button>
```

Figura LVII - Botão Delete

Fonte: (do autor)

A função *deleteEmp* apaga o empregado selecionado da base de dados, grava as alterações, atualiza a *ListEmp* que vai atualizar a *ListView* e seleciona um novo empregado corrente.

```

public void deleteEmp(empregado a)
{
    try
    {
        var este = EmpModel.empregados.Where(x => x.idemp ==
a.idemp).First();
        if (este != null)
        {
            EmpModel.empregados.Remove(este);
            EmpModel.SaveChanges();
            ListEmp = new
ObservableCollection<empregado>(EmpModel.empregados);
            ListEmp.CollectionChanged += ListEmp_CollectionChanged;
            ViewEmp = CollectionViewSource.GetDefaultView(_listEmp);
            ViewEmp.CurrentChanged += ViewEmp_CurrentChanged;
            EmpCorrente = (empregado)ViewEmp.CurrentItem;
            System.Windows.MessageBox.Show("Apagado com Sucesso");
        }
    }
    catch (Exception erro)
    {
        MessageBox.Show(erro.Message);
    }
}

```

O botão *Insert* simplesmente redireciona para a página dois, onde se pode inserir um empregado novo.

```
<Button Content="Insert" DockPanel.Dock="Right" Command="local:Comandos.navegarpag2" HorizontalAlignment="Right" ></Button>
```

Figura LVIII - Botão Insert

Fonte: (do autor)

```

public void navegar(string pag)
{
    switch (pag)
    {
        case "Page1":
            Page1 pag1 = new Page1();
            w.frame.Navigate(pag1);
            break;
        case "Page2":
            Page2 pag2 = new Page2();
            w.frame.Navigate(pag2);
            break;
        case "Page3":
            Page3 pag3 = new Page3();
            w.frame.Navigate(pag3);
            break;
    }
}

```

A interface da página dois é a seguinte.

The screenshot shows a Windows application window titled "Projeto Final". The main title bar has "Projeto Final" and standard window controls. Below the title bar is a menu bar with "Exit" and "Navegação". The main content area has a light blue background and is titled "Inserir Novo Registo". It contains four text input fields with labels and placeholder text:

- *Id: [text box]
- *Nome: [text box]
- *Funcao: [text box]
- *Nº Telemovel: [text box]

Below these fields is a note: "* Campos de preenchimento obrigatorio". At the bottom right are two buttons: "Inserir" and "Carregar Foto".

Figura LIX - Página dois

Fonte: (do autor)

Esta página é constituída por várias *TextBox*, uma *ComboBox* e dois botões.

A *ComboBox* instancia uma classe que permite escolher entre Técnico ou Entregas.

```
public class clsfuncao
{
    public string funcao { get; set; }
}
public List<clsfuncao> Funcao
{
    get
    {
        return new List<clsfuncao>()
        {
            new clsfuncao { funcao="Técnico"},
            new clsfuncao { funcao = "Entregas" },
        };
    }
}
```

O botão Carregar Foto permite carregar uma foto que vai ser usada no registo do empregado.

```
<Button Grid.Column="5" Grid.Row="5" Command="local:Comandos.insertimagem" Margin="0,0,20,0" Width="85" Content="Carregar Foto"/>
```

Figura LX - Botão Carregar Foto

Fonte: (do autor)

```
public void insertimagem(System.Windows.Controls.Image img)
{
    OpenFileDialog dlg = new OpenFileDialog();
    dlg.InitialDirectory = @"C:\Users\Administrator\Desktop";
    dlg.Filter = "*.*|*.JPEG Files (*.jpeg)|*.jpg|*.PNG Files (*.png)|*.png|*.JPG Files (*.jpg)|*.jpg|*.GIF Files (*.gif)|*.gif";
    if (dlg.ShowDialog() == true)
    {
        BitmapImage foto = new BitmapImage();
        foto.BeginInit();
        foto.CacheOption = BitmapCacheOption.OnDemand;
        foto.CreateOptions = BitmapCreateOptions.DelayCreation;
        foto.UriSource = new Uri(dlg.FileName, UriKind.Absolute);
        foto.EndInit();
        img.Source = foto;
    }
}
```

A função insertimagem, abre uma janela em que se pode escolher uma foto, essa foto fica na *Image* img, que mais tarde vai ser usada na inserção de um novo registo.

O botão Inserir permite inserir um novo empregado com os campos presente na página dois.

```
<Button Grid.Column="2" HorizontalAlignment="Right" Grid.Row="5" Command="local:Comandos.inserirEmp" Height="23" Content="Inserir" Width="70"/>
```

Figura LXI - Botão Inserir

Fonte: (do autor)

```
public void inserirEmp(TextBox txtid, TextBox txtnome, ComboBox cmb,
System.Windows.Controls.Image img, TextBox txtTele)
{
    try
    {
        int id;
        long tele;
        if (int.TryParse(txtid.Text, out id))
        {
            if (long.TryParse(txtTele.Text, out tele))
            {
                empregado p = new empregado();
                p.idemp = id;
                p.nome = txtnome.Text;
                p.telefone = tele;
                try
                {
                    p.fotopath = Path.GetFileName(saveFileFromPicture(img, id));
                    var apoio = p.fotopath;
                    try
                    {

```

```

        p.funcao = ((clsfuncao)cmb.SelectedItem).funcao;
        try
        {
            EmpModel.empregados.Add(p);
            EmpModel.SaveChanges();
            ListEmp.Add(p);
            ViewEmp.MoveCurrentToLast();
            EmpCorrente = (empregado)ViewEmp.CurrentItem;
            OnPropertyChanged("ListEmp");
            navegar("Page1");
        }
        catch
        {
            MessageBox.Show("Verificar os Campos");
            reset();
            p.fotopath = apoio;
        }
    }
    catch
    {
        MessageBox.Show("Selecionar Técnico ou Entregas");
    }
}
catch
{
    p.fotopath = null;
    try
    {
        p.funcao = ((clsfuncao)cmb.SelectedItem).funcao;
        try
        {
            EmpModel.empregados.Add(p);
            EmpModel.SaveChanges();
            ListEmp.Add(p);
            ViewEmp.MoveCurrentToLast();
            EmpCorrente = (empregado)ViewEmp.CurrentItem;
            OnPropertyChanged("ListEmp");
            navegar("Page1");
        }
        catch
        {
            MessageBox.Show("Verificar os Campos");
            reset();
        }
    }
    catch
    {
        MessageBox.Show("Selecionar Técnico ou Entregas");
    }
}
else
{
    MessageBox.Show("Verificar Campos");
}
else {
    MessageBox.Show("Verificar Campos");
}

```

```

        }
    }
    catch (Exception error)
    {
        MessageBox.Show(error.Message);
    }
}

```

A função inserirEmp recebe os argumentos inseridos pelo utilizador da aplicação preenchidos no XAML, verifica se estão preenchidos e de acordo as exceções criadas no SQL e adiciona um novo empregado. Após a adição ele atualiza a lista, faz com que o empregado corrente seja o último e redireciona de novo para a página um.

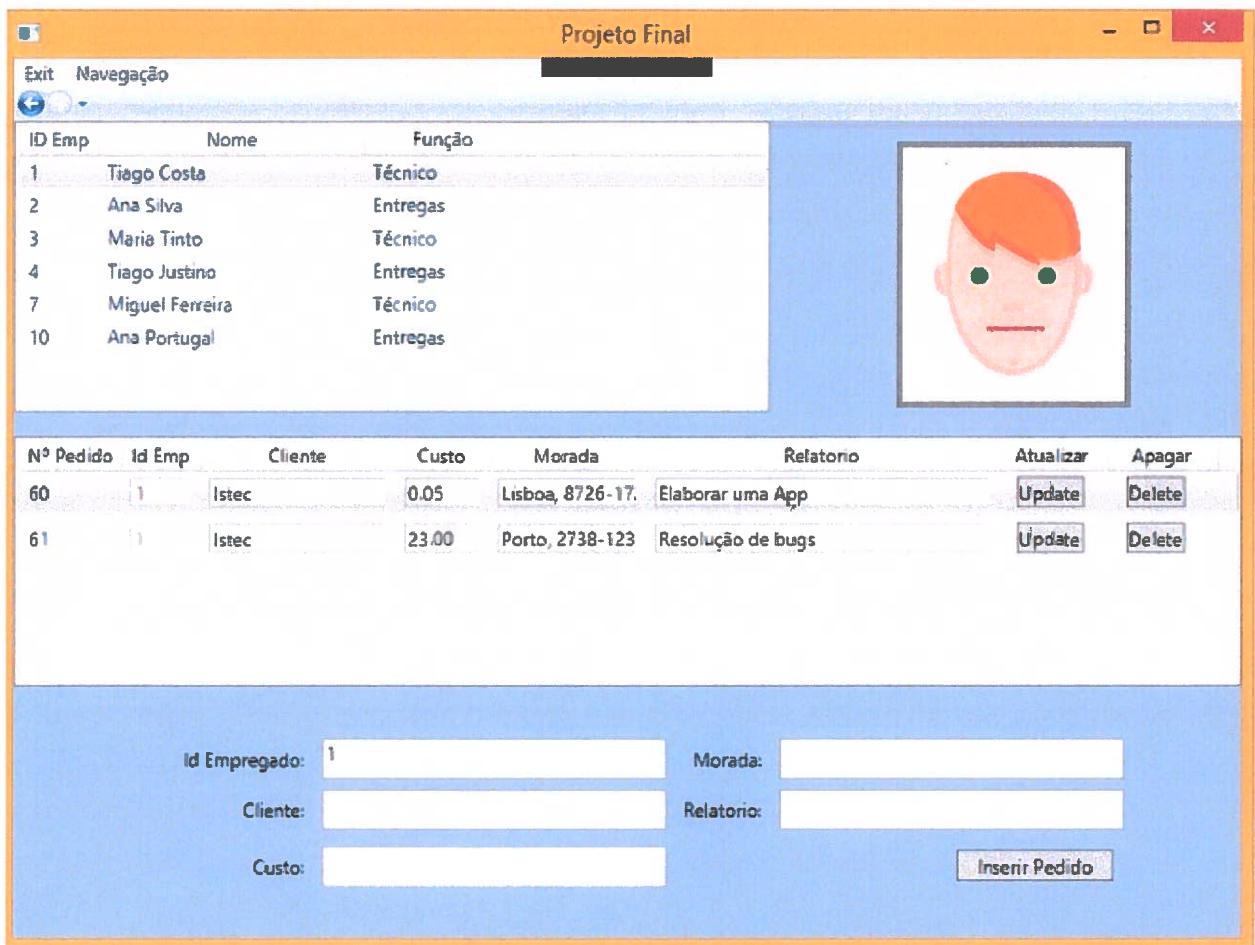


Figura LXII - Page3

Fonte: (do autor)

A página três tem duas *ListView*s, a primeira faz o mesmo que a *ListView* da página um, apresenta todos os empregados, mas neste caso não apresenta o número de telemóvel.

A segunda *ListView* apresenta os pedidos de cada empregado, sendo possível apagar ou atualizar o respetivo pedido.

Também é possível inserir novos pedidos, basta preencher as *TextBox* com os dados do pedido.

Para a página três foi criada uma classe nova por motivos de organização, a classe designa-se por *ViewModelPedidos*.

Repete-se o processo que foi feito na página um e na *MainWindow* para apresentar a informação nas *listViews*.

```
<Page.Resources>
    <local:ViewModelPedidos x:Key="bd"></local:ViewModelPedidos>
    <local:converte x:Key="top"></local:converte>
    <local:DecimalConverter x:Key="decimalconverter"></local:DecimalConverter>
</Page.Resources>
<Grid DataContext="{Binding Source={StaticResource bd}}>
```

Figura LXIII - XAML Page3 ViewModelPedidos

Fonte: (do autor)

Esta instanciado no XAML a classe *ViewModelPedidos*, onde esta o código que permite trabalhar com a base de dados, a classe converte, permite obter a imagem a partir do caminho onde esta guardada e a classe *DecimalConverter*, é uma classe que transforma as vírgulas em pontos finais, foi usada no custo por motivo de guardar o valor decimal no SQL.

```
<ListView Grid.Row="0" ItemsSource="{Binding ListEmp}" Name="lst" Margin="0,0,300,0" IsSynchronizedWithCurrentItem="True" >
<ListView.View>
    <GridView>
        <GridViewColumn Header="ID Emp" DisplayMemberBinding="{Binding idemp}" Width="50"/>
        <GridViewColumn Header="Nome" DisplayMemberBinding="{Binding nome}" Width="170"/>
        <GridViewColumn Header="Função" DisplayMemberBinding="{Binding funcao}" Width="100" />
    </GridView>
</ListView.View>
```

Figura LXIV - ListView empregados

Fonte: (do autor)

```
ListEmp = new ObservableCollection<empregado>(bd.empregados.Include("pedidos").ToList());
ViewEmp = CollectionViewSource.GetDefaultView(ListEmp);
ViewEmp.CurrentChanging += ViewEmp_CurrentChanging;
EmpCorrente = ViewEmp.CurrentItem as empregado;
```

Figura LXV – Listar pedidos do empregado

Fonte: (do autor)

Código da *ListView* dos pedidos do empregado selecionado.

```
<ListView Margin="0,15,0,0" Grid.Row="1" Name="lstntas" IsSynchronizedWithCurrentItem="True"
ItemsSource="{Binding SelectedItem.pedidos, ElementName=lst}">
    <ListView.View>
        <GridView>
            <GridViewColumn Header="Nº Pedido" Width="65"
DisplayMemberBinding="{Binding idpedido}"/>

            <GridViewColumn Header="Id Emp" Width="50" >
                <GridViewColumn.CellTemplate>
                    <DataTemplate>
                        <TextBox TextAlignment="Left" Text="{Binding idemp}"
Width="50" HorizontalAlignment="Left" IsEnabled="False" ></TextBox>
                    </DataTemplate>
                </GridViewColumn.CellTemplate>
            </GridViewColumn>

            <GridViewColumn Header="Cliente" Width="80" >
                <GridViewColumn.CellTemplate>
                    <DataTemplate>
                        <TextBox HorizontalAlignment="Left" Text="{Binding nomecliente}"
TextAlignment="Left" Width="175" ></TextBox>
                    </DataTemplate>
                </GridViewColumn.CellTemplate>
            </GridViewColumn>

            <GridViewColumn Header="Custo" Width="60" >
                <GridViewColumn.CellTemplate>
                    <DataTemplate>
                        <TextBox TextAlignment="Left" Text="{Binding custo,
Converter={StaticResource decimalconverter}}" HorizontalAlignment="Left" Width="75"
></TextBox>
                    </DataTemplate>
                </GridViewColumn.CellTemplate>
            </GridViewColumn>

            <GridViewColumn Header="Morada" Width="150" >
                <GridViewColumn.CellTemplate>
                    <DataTemplate>
                        <TextBox HorizontalAlignment="Left" Text="{Binding morada}"
TextAlignment="Left" Width="400" ></TextBox>
                    </DataTemplate>
                </GridViewColumn.CellTemplate>
            </GridViewColumn>

            <GridViewColumn Header="Relatorio" Width="225" >
                <GridViewColumn.CellTemplate>
                    <DataTemplate>
                        <TextBox TextAlignment="Left" Text="{Binding relatorio}"
HorizontalAlignment="Left" Width="400" ></TextBox>
                    </DataTemplate>
                </GridViewColumn.CellTemplate>
            </GridViewColumn>

            <GridViewColumn Header="Atualizar" Width="70">
                <GridViewColumn.CellTemplate>
                    <DataTemplate>
                        <StackPanel Margin="6,2,6,2">
                            <Button Content="Update"
Command="local:Comandos.updatepedido" />
                    </DataTemplate>
                </GridViewColumn.CellTemplate>
            </GridViewColumn>
        </GridView>
    </ListView.View>
</ListView>
```

```

        </StackPanel>
    </DataTemplate>
</GridViewColumn.CellTemplate>

</GridViewColumn>

<GridViewColumn Header="Apagar" Width="70">
    <GridViewColumn.CellTemplate>
        <DataTemplate>
            <StackPanel Margin="6,2,6,2">
                <Button Content="Delete"
Command="local:Comandos.deletepedido"/>
            </StackPanel>
        </DataTemplate>
    </GridViewColumn.CellTemplate>

</GridViewColumn>

</GridView>
</ListView.View>

</ListView>

```

Na primeira *ListView*, vai popular com os empregados e os campos Id, nome e função.

Na segunda *ListView*, vai mostrar todos os pedidos do empregado selecionado, com os campos, *idpedido*, *idemp*, nomecliente, morada, custo, relatório, um botão para apagar e outro para atualizar por cada registo de um pedido, só é possível pelo uso do *DataTemplate*.

```

public void apagarpedido(System.Windows.Controls.ListView lst)
{
    using (bd = new istecEntities())
    {
        var pedido = ((pedido)lst.SelectedItem);
        var este = bd.pedidos.Where(x => x.idemp == pedido.idemp && x.idpedido == pedido.idpedido).FirstOrDefault();

        if (este != null)
        {
            try {
                var p = este.empregado;
                p.pedidos.Remove(este);
                bd.SaveChanges();
                OnPropertyChanged("ListEmp");
                ListEmp = new ObservableCollection<empregado>(bd.empregados.Include("pedidos").ToList());
                ViewEmp = CollectionViewSource.GetDefaultView(ListEmp);
                ViewEmp.CurrentChanging += ViewEmp_CurrentChanging;
                ViewEmp.MoveCurrentTo(p);
                EmpCorrente = ViewEmp.CurrentItem as empregado;
                System.Windows.MessageBox.Show("Apagado com Sucesso");
            }
            catch (Exception erro)
            {
                MessageBox.Show(erro.Message);
            }
        }
    }
}

```

Figura LXVI - Função apagarpedido

Fonte: (do autor)

A função apagarpedido é executada quando se pressiona o botão *Delete*, caso consiga ela apaga o pedido selecionado e todos os campos referentes ao mesmo da base de dados, atualiza a *ListView* e mostra o mesmo empregado que estava selecionado e os seus pedidos, se não mostra uma mensagem de erro.

```
public void updatepedido(ListView lst)
{
    using (bd = new istecEntities())
    {
        var pedido = ((pedido)lst.SelectedItem);
        var este = bd.pedidos.Where(x => x.idemp == pedido.idemp && x.idpedido == pedido.idpedido).FirstOrDefault();
        if (este != null)
        {
            var p = este.empregado;
            este.idemp = pedido.idemp;
            este.nomecliente = pedido.nomecliente;
            //este.idpedido = pedido.idpedido;
            este.custo = Math.Round(pedido.custo, 2);
            este.morada = pedido.morada;
            este.relatorio = pedido.relatorio;
            bd.SaveChanges();
            OnPropertyChanged("ListEmp");
            ListEmp = new ObservableCollection<empregado>(bd.empregados.Include("pedidos").ToList());
            ViewEmp = CollectionViewSource.GetDefaultView(ListEmp);
            ViewEmp.CurrentChanging += ViewEmp_CurrentChanging;
            ViewEmp.MoveCurrentTo(p);
            EmpCorrente = ViewEmp.CurrentItem as empregado;
            // teste();
            System.Windows.MessageBox.Show("Atualizado com Sucesso");
        }
        else { MessageBox.Show("Verificar os Campos"); }
    }
}
```

Figura LXVII - Função updatepedido

Fonte: (do autor)

Quando pressionado o botão *Update* é executada a função updatepedido. Esta função atualiza todos os campos do pedido selecionado, nomeadamente, *idemp*, *nomecliente*, *custo*, *morada* e *relatório*. Se permitir atualizar, os dados são atualizados na base de dados, a *Listview* é atualizada por causa do *ListEmp* também é atualizado e coloca o pedido que foi atualizado em primeiro lugar, caso mude de página ou reiniciar aplicação a ordem dos pedidos está de acordo o *Id* do pedido e crescente, foi escolhida esta opção caso tivesse muitos pedidos e quisesse confirmar se realmente foi alterado.

```
public void inserirpedido(TextBox idempTextBox, TextBox clientTextBox, TextBox custoTextBox,
    TextBox moradaTextBox, TextBox relatorioTextBox)
{
    using (bd = new istecEntities())
    {
        int idem;
        decimal custo;
```

A função `inserirpedido`, está associado ao botão Inserir Pedido. Quando executada, ela insere um novo pedido com os campos preenchidos pelo utilizador da aplicação. Faz a verificação se os campos estão corretamente preenchidos e avisa com uma `MessageBox` caso não estejam. O pedido está sempre associado a um empregado, pois o `idemp` faz essa ligação, visto que é uma chave estrangeira numa tabela e na outra é a chave principal, tem sempre um `idemp` nas duas tabelas, isto caso o

empregado tenha algum pedido. Depois de inserir um pedido no respetivo empregado, a função atualiza o *ListEmp* que por si vai atualizar as *ListView*s e o empregado selecionado é o que foi inserido um pedido novo.

Conclusão

A elaboração do Projeto Global foi possível ao conhecimento adquirido nos três anos de licenciatura e num ano de pós-graduação, tendo de destacar as unidades curriculares de programação e virtualização.

Em função do objetivo de estudo, posso concluir que a virtualização é uma solução considerada pelas empresas, pois é uma solução em que não se tem de investir grandes quantias em infraestruturas, manutenção, mão de obra ou sustentabilidade em relação ao *IT* convencional.

A aplicação elaborada permite demonstrar que, até em pequenas e médias empresas, a capacidade de dispor de uma solução robusta e escalável é cada vez mais uma necessidade, visto que os negócios dependem mais das tecnologias colocadas à disposição dos mesmos.

Com a realização deste projeto, consegui enaltecer os meus conhecimentos nas áreas de programação e virtualização, tendo assim, conseguido concluir o projeto a cumprir com os objetivos propostos.

Referências Bibliográficas

- Azure Cloud.* (23 de Maio de 2018). Obtido de <https://azure.microsoft.com/en-in/overview/what-is-cloud-computing/> (consultado em 20/05/2018)
- Cloud Computing.* (5 de Outubro de 2017). Obtido de <https://pt.linkedin.com/pulse/o-que-%C3%A9-cloud-computing-entenda-sua-defini%C3%A7%C3%A3o-e-marcos-davila-mba> (consultado em 20/05/2018)
- Desmond, B., Richards, J., Allen, R., & Lowe-Norris, A. G. (2013). Active Directory: Designing, Deploying, and Running Active Directory. O'Reilly Media.
- Entity Framework.* (30 de Maio de 2018). Obtido de <http://www.entityframeworktutorial.net/what-is-entityframework.aspx> (consultado em 26/05/2018)
- IBM.* (2018). Obtido de <https://www.ibm.com/cloud/learn/what-is-cloud-computing> (consultado em 20/05/2018)
- Mell, P., & Grance, T. (Setembro de 2011). *NIST*. Obtido de <https://nvlpubs.nist.gov/nistpubs/legacy/sp/nistspecialpublication800-145.pdf> (consultado em 20/05/2018)
- Microsoft.* (22 de Maio de 2018). Obtido de <https://azure.microsoft.com/en-us/overview/what-is-a-virtual-machine/> (consultado em 16/05/2018)
- Migliore, M. (26 de Junho de 2011). Obtido de <https://code.msdn.microsoft.com/How-to-implement-MVVM-71a65441> (consultado em 26/05/2018)
- Minasi, M., Greene, K., Booth, C., Butler, R., McCabe, J., Panek, R., . . . Roth, S. (2013). Mastering Windows Server 2012 R2. Sybex.
- Mistry, R., & Misner, S. (2013). Introducing Microsoft SQL Server 2014 Technical Overview. Microsoft Press.
- MVVM.* (10 de Abril de 2012). Obtido de <https://addyosmani.com/blog/understanding-mvvm-a-guide-for-javascript-developers/> (consultado em 26/05/2018)
- Open Source.* (22 de Maio de 2018). Obtido de <https://opensource.com/resources/virtualization> (consultado em 16/05/2018)
- Red Hat.* (22 de Maio de 2018). Obtido de <https://www.redhat.com/en/topics/virtualization> (consultado em 16/05/2018)
- Tulloch, M. (2013). Introducing Windows Server 2012 R2. Microsoft Press.

Visual Studio. (23 de Maio de 2018). Obtido de <https://msdn.microsoft.com/en-us/library/dn762121.aspx> (consultado em 26/05/2018)

VMware. (22 de Maio de 2018). Obtido de <https://www.vmware.com/solutions/virtualization.html> (consultado em 16/05/2018)

Anexos

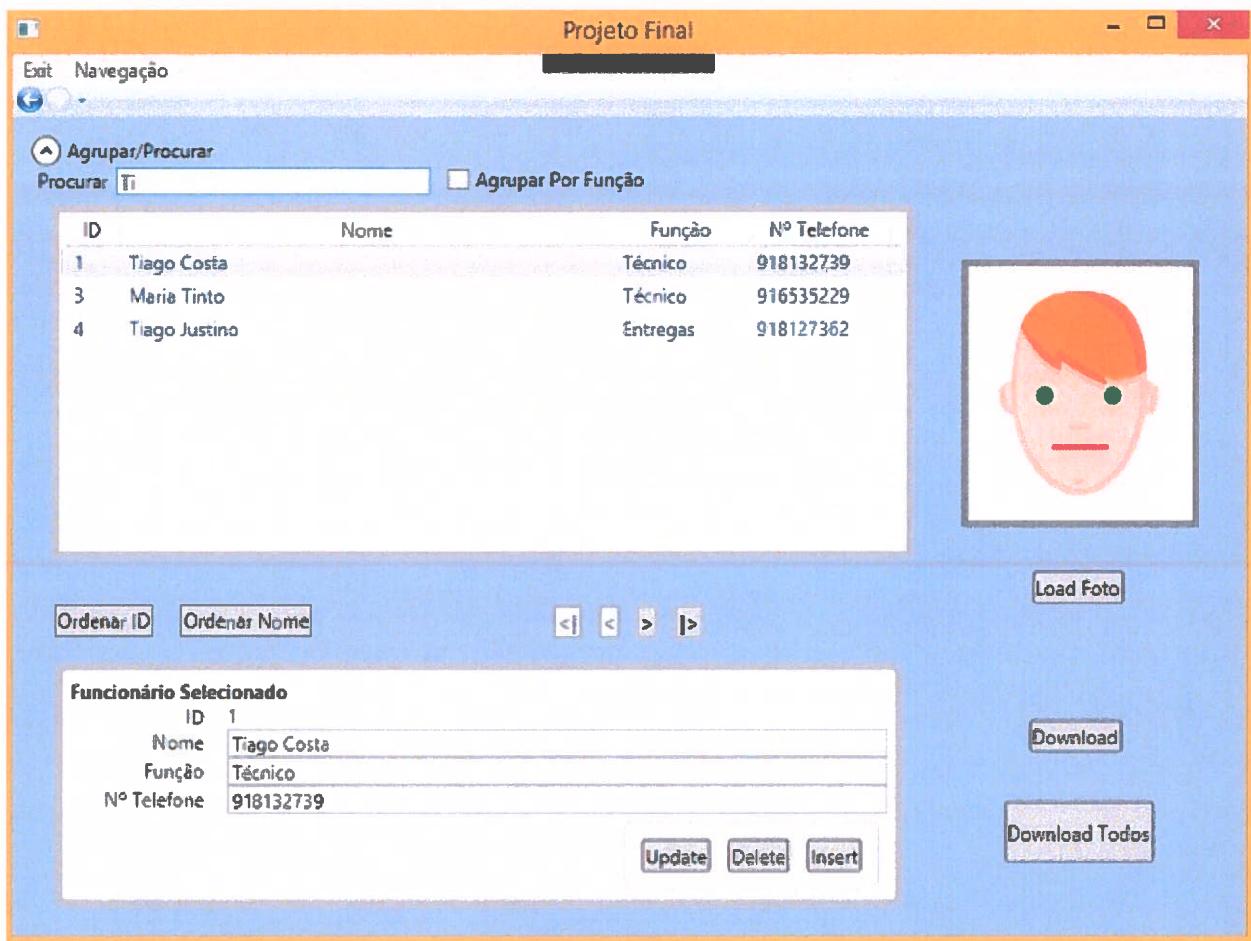


Figura LXVIII - Filtrar o nome

Fonte: (do autor)

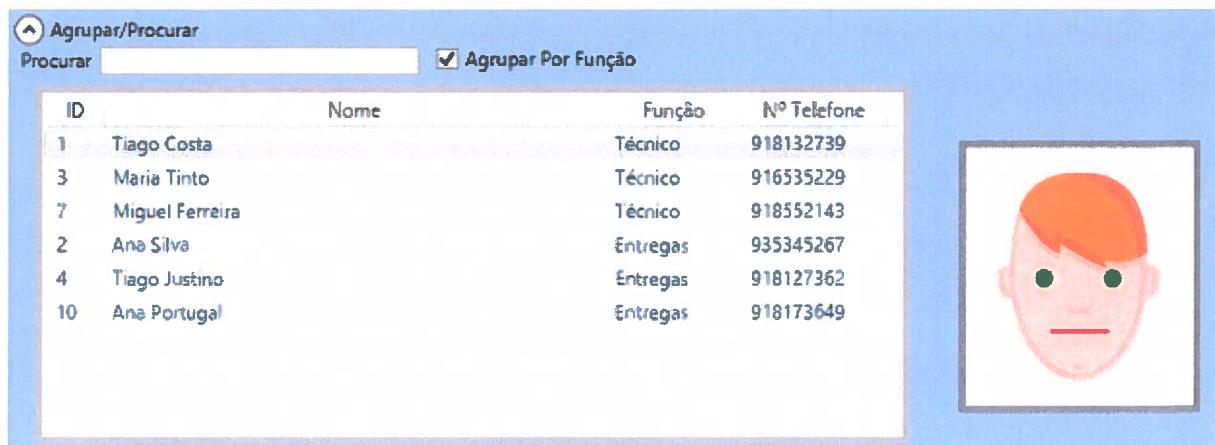


Figura LXIX - Agrupar por função

Fonte: (do autor)

ID	Nome	Função	Nº Telefone
10	Ana Portugal	Entregas	918173649
7	Miguel Ferreira	Técnico	918552143
4	Tiago Justino	Entregas	918127362
3	Maria Tinto	Técnico	916535229
2	Ana Silva	Entregas	935345267
1	Tiago Costa	Técnico	918132739

Figura LXX - Ordenar Id descendente

Fonte: (do autor)

ID	Nome	Função	Nº Telefone
10	Ana Portugal	Entregas	918173649
2	Ana Silva	Entregas	935345267
3	Maria Tinto	Técnico	916535229
7	Miguel Ferreira	Técnico	918552143
1	Tiago Costa	Técnico	918132739
4	Tiago Justino	Entregas	918127362

Figura LXXI - Ordenar nome ascendente

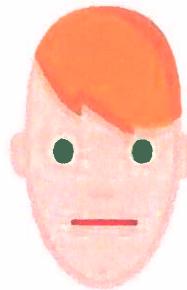
Fonte: (do autor)

ID	Nome	Função	Nº Telefone
4	Tiago Justino	Entregas	918127362
1	Tiago Costa	Técnico	918132739
7	Miguel Ferreira	Técnico	918552143
3	Maria Tinto	Técnico	916535229
2	Ana Silva	Entregas	935345267
10	Ana Portugal	Entregas	918173649

Figura LXXII - Ordenar nome descendente

Fonte: (do autor)

Registo do Funcionario



Id: 1 Nome: Tiago Costa

Pedidos do Funcionario					
Nº Pedido	IdEmp	Nome Cliente	Custo	Morada	Relatorio
60	1	Istec	120,00	Lisboa, 8726-172	Elaborar uma App
61	1	Istec	23,00	Porto, 2738-123	Resolução de bugs

Figura LXXIII - Pdf individual

Fonte: (do autor)

ViewModel.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.ComponentModel;
using System.Collections.ObjectModel;
using System.Windows.Data;
using System.Windows.Controls;
using Microsoft.Win32;
using System.Data.Entity;
using System.Runtime.InteropServices;
using Microsoft.Win32.SafeHandles;
using System.Collections.Specialized;
using System.Windows.Media.Imaging;
using System.Windows;
using System.Data.Entity.Infrastructure;
using System.IO;
using System.Windows.Media;
using iTextSharp;
using iTextSharp.text;
using iTextSharp.text.pdf;
```

```

using iTextSharp.text.pdf.draw;
using System.Diagnostics;

namespace FinalProject
{
    public class ViewModel : INotifyPropertyChanged
    {
        #region Colecções e Dados
        public MainWindow w { get; set; }
        //Entities
        istecEntities _empModel;
        public istecEntities EmpModel // criação de um modelo onde contem as informações
dos empregados
        {
            get { return _empModel; }
            set {
                _empModel = value;
                OnPropertyChanged("empModel");
            }
        }

        //ObservableCollection
        private ObservableCollection<empregado> _listEmp;
        public ObservableCollection<empregado> ListEmp //construtor do observablecollection
        {
            get { return _listEmp; }
            set {
                _listEmp = value;
                OnPropertyChanged("ListEmp");
            }
        }

        //CollectionView
        public ICollectionView ViewEmp { get; set; } //construtor do icollection
        private empregado _empCorrente;
        public empregado EmpCorrente
        {
            get { return _empCorrente; }
            set {
                _empCorrente = value;
                OnPropertyChanged("EmpCorrente");
            }
        }

        #endregion
        #region PropertyChanged
        public event PropertyChangedEventHandler PropertyChanged;
        public void OnPropertyChanged(String nome)
        {
            if (PropertyChanged != null) PropertyChanged(this, new
PropertyChangedEventArgs(nome));
        }
        #endregion

        private void ViewEmp_CurrentChanged(object sender, EventArgs e)
        {
            EmpCorrente = (empregado)ViewEmp.CurrentItem;
        }
        private void ListEmp_CollectionChanged(object sender,
NotifyCollectionChangedEventArgs e)
        {
            OnPropertyChanged("ListEmp");
        }
    }
}

```

```

public ViewModel() //construtor da classe
{
    w = (MainWindow)Application.Current.MainWindow;
    EmpModel = new istecEntities();
    //listar os empregados e coloca na ListView
    ListEmp = new ObservableCollection<empregado>(EmpModel.empregados);
    ViewEmp = CollectionViewSource.GetDefaultView(_listEmp);
    ListEmp.CollectionChanged += ListEmp_CollectionChanged;
    ViewEmp.CurrentChanged += ViewEmp_CurrentChanged;
    //identifica o empregado selecionado
    EmpCorrente = (empregado)ViewEmp.CurrentItem;
}

void reset()
{
    EmpModel = new istecEntities();
}
void reset2()
{
    ListEmp = new
ObservableCollection<empregado>(EmpModel.empregados.Include("pedidos").ToList());
}
#region Comandos

    public void sair()//fecha o programa
{
    Application.Current.Shutdown();
}

    public bool canGoFirst()//verifica se a posição do empregado corrente é maior que 0
para poder ir para a primeira posição
{
    return ViewEmp.CurrentPosition > 0;
}
    public bool canGoLast() //verifica se a posição do empregado corrente é menor que a
penultima posição da lista
{
    return ViewEmp.CurrentPosition < (ViewEmp.Cast<empregado>().Count() - 1);
}
    public void gofirst()//vai para a primeira posição se o cangofirst() retornar true
{
    ViewEmp.MoveCurrentToFirst();
    EmpCorrente = (empregado)ViewEmp.CurrentItem;
}
    public void goLast()// vai para a ultima posição se o cangolast() retornar true
{
    ViewEmp.MoveCurrentToLast();
    EmpCorrente = (empregado)ViewEmp.CurrentItem;
}
    public void goNext()//vai para o proximo empregado
{
    ViewEmp.MoveCurrentToNext();
    EmpCorrente = (empregado)ViewEmp.CurrentItem;
}
    public void goPrevious()//vai para o empregado anterior
{
    ViewEmp.MoveCurrentToPrevious();
    EmpCorrente = (empregado)ViewEmp.CurrentItem;
}

    public void navegar(string pag)//permite navegar entre as paginas
{

```

```

        switch (pag)
    {
        case "Page1":
            Page1 pag1 = new Page1();
            w.frame.Navigate(pag1);
            break;
        case "Page2":
            Page2 pag2 = new Page2();
            w.frame.Navigate(pag2);
            break;
        case "Page3":
            Page3 pag3 = new Page3();
            w.frame.Navigate(pag3);
            break;

    }
}

#region Sort
private bool _directionAscending = true;//variavel de controlo
public void ordenarid()
{
    if (_directionAscending == true)
    {
        _directionAscending = false;

    }
    else
    {
        _directionAscending = true;
    }
    //se _directionAscending for verdadeiro, a lista é ascendente se nao e
descendente
    ListSortDirection direccao = (_directionAscending) ? ListSortDirection.Ascending
: ListSortDirection.Descending;
    using (ViewEmp.DeferRefresh())
    {
        ViewEmp.SortDescriptions.Clear();
        ViewEmp.SortDescriptions.Add(new SortDescription("idemp", direccao));//ordena
pelo idemp
    }
}

private bool _direAs = false; // variavel de controlo
public void ordenarnome()
{
    if (_direAs == true)
    {
        _direAs = false;

    }
    else
    {
        _direAs = true;
    }
    // se _direAs for verdadeiro, a lista é ascendente se nao e descendente
}

```

```

        ListSortDirection direccao = (_direAs) ? ListSortDirection.Ascending :
ListSortDirection.Descending;
        using (ViewEmp.DeferRefresh())
        {
            ViewEmp.SortDescriptions.Clear();

            ViewEmp.SortDescriptions.Add(new SortDescription("nome", direccao));// ordena
pelo nome

        }
#endregion
#region Filtro

private string _searchText;
public string SearchText
{
    get { return _searchText; }
    set
    {
        _searchText = value;
        ViewEmp.Filter = FilterData;
        OnPropertyChanged("SearchText");//valor que vai ser procurado
    }
}

private bool FilterData(object item) //filtra a lista e verifica se tem o valor
procurado
{
    var value = (empregado)item;
    if (value == null || value.nome == null)
        return false;

    return value.nome.Contains(SearchText);
}

#endregion
#region Agrupar
private bool _groupByClube;
public bool GroupByClube
{
    get { return _groupByClube; }
    set
    {
        _groupByClube = value;
        GroupData();
        OnPropertyChanged("GroupByClube");
    }
}
private void GroupData() // agrupa os empregado por função
{
    using (ViewEmp.DeferRefresh())
    {
        ViewEmp.GroupDescriptions.Clear();
        if (_groupByClube) ViewEmp.GroupDescriptions.Add(new
PropertyGroupDescription("funcao"));
    }
}
#endregion

```

```

#endregion
#region BD

public void updateEmp(empregado a) //atualizar o empregado
{
    try
    {
        var este = EmpModel.empregados.First(x => x.idemp == a.idemp); //verifica o
        empregado pelo id e vai buscar todos os seus dados
        if (este != null)
        {
            //recebe os dados preenchidos
            este.idemp = a.idemp;
            este.nome = a.nome;
            este.funcao = a.funcao;
            este.telemovel = a.telemovel;
            if (a.fotopath != null) este.fotopath = a.fotopath;
            if (este.nome != "" && este.funcao != "")
            {
                EmpModel.SaveChanges(); //grava na base de dados
                MessageBox.Show("Atualizado com Sucesso");
            }
            else MessageBox.Show("Todos os campos são de preenchimento obrigatorio");
        }
    }
    catch
    {
        MessageBox.Show("Verificar os Campos");
    }
}

public void deleteEmp(empregado a) //apagar empregado
{
    try
    {
        var este = EmpModel.empregados.Where(x => x.idemp == a.idemp).First();
        //verifica o empregado pelo id e vai buscar todos os seus dados
        if (este != null)
        {
            EmpModel.empregados.Remove(este); //apaga o empregado e todos os seus
            dados
            EmpModel.SaveChanges(); //grava na base de dados
            //atualiza a lista e o empregado corrente
            ListEmp = new ObservableCollection<empregado>(EmpModel.empregados);
            ListEmp.CollectionChanged += ListEmp_CollectionChanged;
            ViewEmp = CollectionViewSource.GetDefaultView(_listEmp);
            ViewEmp.CurrentChanged += ViewEmp_CurrentChanged;
            EmpCorrente = (empregado)ViewEmp.CurrentItem;
            System.Windows.MessageBox.Show("Apagado com Sucesso");
        }
    }
    catch (Exception erro)
    {
        MessageBox.Show(erro.Message);
    }
}

```

```

    public void inserirEmp(TextBox txtid, TextBox txtnome, ComboBox cmb,
System.Windows.Controls.Image img, TextBox txtTele)//inserir novo empregado
{
    //recebe os valores inseridos pelo utilizador e testa para ver se foram
corretamente preenchidos
    try
    {
        int id;
        long tele;
        if (int.TryParse(txtid.Text, out id))
        {
            if (long.TryParse(txtTele.Text, out tele))
            {
                empregado p = new empregado();

                p.idemp = id;
                p.nome = txtnome.Text;
                p.telemovel = tele;
                try
                {
                    p.fotopath = Path.GetFileName(saveFileFromPicture(img, id));///
vai buscar a imagem a funcao saveFileFromPicture
                    var apoio = p.fotopath;
                    try
                    {
                        p.funcao = ((clsfuncao)cmb.SelectedItem).funcao;
                        try
                        {
                            EmpModel.empregados.Add(p); // adiciona o empregado
                            EmpModel.SaveChanges(); //garava na base de dados
                            ListEmp.Add(p); //adiciona a lista
                            ViewEmp.MoveCurrentToLast(); // muda para o ultimo
empregado(o que foi criado)
                            EmpCorrente = (empregado)ViewEmp.CurrentItem; //atualiza o
empregado corrente
                            onPropertyChanged("ListEmp");
                            navegar("Page1");
                        }
                        catch
                        {
                            MessageBox.Show("Verificar os Campos");
                            reset();
                            p.fotopath = apoio;
                        }
                    }
                    catch
                    {
                        MessageBox.Show("Selecionar Técnico ou Entregas");
                    }
                }
                catch
                {
                    p.fotopath = null;
                    try
                    {
                        p.funcao = ((clsfuncao)cmb.SelectedItem).funcao;
                        try
                        {

```

```

        EmpModel.empregados.Add(p); // adiciona o empregado
        EmpModel.SaveChanges(); // garava na base de dados
        ListEmp.Add(p); // adiciona a lista
        ViewEmp.MoveCurrentToLast(); // muda para o ultimo
    }

    empregado(o que foi criado)
    empregado corrente

    onPropertyChanged("ListEmp");
    navegar("Page1");

}

catch
{
    MessageBox.Show("Verificar os Campos");
    reset();
}

}

catch
{
    MessageBox.Show("Selecionar Técnico ou Entregas");
}

}

else
{
    MessageBox.Show("Verificar Campos");
}

else {
    MessageBox.Show("Verificar Campos");
}

}

catch (Exception error)
{
    MessageBox.Show(error.Message);
}
}

public void loadimagem(System.Windows.Controls.Image img, empregado a) // atualiza a
imagem do empregado selecionado
{
    string oldfich = "";
    OpenFileDialog dlg = new OpenFileDialog();
    dlg.InitialDirectory = @"C:\Users\Administrator\Desktop";
    dlg.Filter = "*.*|*.|JPEG Files (*.jpeg)|*.jpeg|PNG Files (*.png)|*.png|JPG
Files (*.jpg)|*.jpg|GIF Files (*.gif)|*.gif";
    if (dlg.ShowDialog() == true) // abre uma janela do windows onde podemos
selecionar a foto
    {
        try
        {
            string fich = Environment.CurrentDirectory;
            fich = fich.Substring(0, fich.IndexOf("bin")) + "imagens\\" +
EmpCorrente.idemp.ToString() + ".png";
            if (EmpCorrente.fotopath != null && EmpCorrente.fotopath ==
Path.GetFileName(fich))
            {

```

```

        oldfich = fich;
        fich = fich.Replace(@".", @"B.");
    }
    EmpCorrente.fotopath = Path.GetFileName(fich); //a foto do empregado
corrente é igual ao caminho do ficheiro

        File.Copy(dlg.FileName, fich, true);
        OnPropertyChanged("EmpCorrente");
        updateEmp(EmpCorrente);
    }
    catch (Exception erro) { MessageBox.Show(erro.Message); }
}
}

public string saveFileFromPicture(System.Windows.Controls.Image img, int id) // guarda a imagem que vai ser usada no inserir
{
    string fich = Environment.CurrentDirectory;
    fich = fich.Substring(0, fich.IndexOf("bin")) + "imagens\\" + id.ToString() +
".png";
    //RenderTargetBitmap bitmap = new RenderTargetBitmap((int)img.ActualWidth,
(int)img.ActualHeight, 150, 170, PixelFormats.Pbgra32);
    RenderTargetBitmap bitmap = new RenderTargetBitmap((int)img.Source.Width,
(int)img.Source.Height, 96, 96, PixelFormats.Pbgra32);
    bitmap.Render(img);
    BitmapFrame frame = BitmapFrame.Create(bitmap);
    var encoder = new PngBitmapEncoder();
    encoder.Frames.Add(frame);
    using (var stream = File.Create(fich))
    {
        encoder.Save(stream);
    }
    return fich;
}

public void insertimagem(System.Windows.Controls.Image img) // insere a imagem e guarda no img.Source
{
    OpenFileDialog dlg = new OpenFileDialog();
    dlg.InitialDirectory = @"C:\Users\Administrator\Desktop";
    dlg.Filter = "*.*|*.JPEG Files (*.jpeg)|*.jpg|*.PNG Files (*.png)|*.png|*.JPG
Files (*.jpg)|*.gif|*.GIF Files (*.gif)|*.gif";
    if (dlg.ShowDialog() == true)//abre uma janela do windows onde podemos selecionar
a foto
    {
        BitmapImage foto = new BitmapImage();
        foto.BeginInit();
        foto.CacheOption = BitmapCacheOption.OnDemand;
        foto.CreateOptions = BitmapCreateOptions.DelayCreation;
        //foto.DecodePixelHeight = 255;
        //foto.DecodePixelWidth = 255;
        foto.UriSource = new Uri(dlg.FileName, UriKind.Absolute); //nome e caminho da
        imagem
        foto.EndInit();
        img.Source = foto;//foto "guardada" no img
    }
}

```

```

}
#endifregion
#region Função
public class clsfuncao
{
    public string funcao { get; set; }
}
public List<clsfuncao> Funcao //a combox tem a opção de escolher entre tecnico e entregas
{
    get
    {
        return new List<clsfuncao>()
            new clsfuncao { funcao="Técnico"},
            new clsfuncao { funcao = "Entregas" },
        };
    }
}
#endifregion

#region PDF
public void guardaPdf(int num) //cria um pdf na pasta documentos com o nome do id do empregado, ex: 1.png, que contem todos os seus pedidos
{
    string caminho = Environment.CurrentDirectory;
    caminho = caminho.Substring(0, caminho.IndexOf("bin")) + "documentos\\" +
num.ToString() + ".pdf"; //deteta o caminho
    String fotopath = (EmpCorrente.fotopath != null) ?
Environment.CurrentDirectory.Substring(0, Environment.CurrentDirectory.IndexOf("bin")) +
"imagens\\" + EmpCorrente.fotopath :
Environment.CurrentDirectory.Substring(0,
Environment.CurrentDirectory.IndexOf("bin")) + "imagens\\nofoto.png"; //se existir um photopath do empregado corrente, ele vai buscar esse caminho, se nao vai buscar o caminho do nofoto(foto atribuida ao empregados sem foto)
    Document doc = new Document(PageSize.A4, 15f, 15f, 15f, 15f);
    try
    {
        // Escrita integral no documento
        FileStream fs = new FileStream(caminho, FileMode.Create, FileAccess.Write);
        PdfWriter w = PdfWriter.GetInstance(doc, fs);
        doc.Open();
        Font titulo1 = FontFactory.GetFont("COURIER", 20f);
        titulo1.SetStyle(Font.BOLD);
        Paragraph p1 = new Paragraph("Registo do Funcionario\n", titulo1);
        p1.Alignment = Element.ALIGN_CENTER;
        doc.Add(p1);
        doc.Add(new Phrase("\n\n"));
        var foto = iTextSharp.text.Image.GetInstance(fotopath); // imagem do
empregado
        foto.Alignment = iTextSharp.text.Image.ALIGN_CENTER;
        foto.ScaleAbsoluteHeight(200);
        foto.ScaleAbsoluteWidth(170);
        doc.Add(foto);
        doc.Add(new Phrase("\n\n\n"));
        Font titulo2 = FontFactory.GetFont("COURIER", 15f);
        Paragraph p2 = new Paragraph("Id: "+EmpCorrente.idemp.ToString() + " Nome: "
" + EmpCorrente.nome, titulo2); // Id e nome do empregado corrente
        p2.IndentationLeft = 120;
        p2.SetLeading(5.0f, 4.0f);
        doc.Add(p2);
        doc.Add(new Phrase("\n\n"));
        PdfPTable table = new PdfPTable(6);

```

```

        table.DefaultCell.BorderColor = BaseColor.BLACK;
        table.DefaultCell.Border = Rectangle.BOX;
        table.DefaultCell.HorizontalAlignment = Element.ALIGN_LEFT;
        PdfPCell cell = new PdfPCell(new Phrase("Pedidos do Funcionario", titulo1));
        cell.Colspan = 6;
        cell.HorizontalAlignment = Element.ALIGN_CENTER;
        cell.VerticalAlignment = Element.ALIGN_CENTER;
        table.AddCell(cell);
        table.AddCell(new Phrase("Nº Pedido"));
        table.AddCell(new Phrase("IdEmp"));
        table.AddCell(new Phrase("Nome Cliente"));
        table.AddCell(new Phrase("Custo"));
        table.AddCell(new Phrase("Morada"));
        table.AddCell(new Phrase("Relatorio"));
        foreach (pedido n in EmpCorrente_pedidos) //ciclo que corre todos os pedidos
do empregado corrente e insere numa tabela
        {
            table.AddCell(new Phrase(n.idpedido.ToString()));
            table.AddCell(new Phrase(n.idemp.ToString()));
            table.AddCell(new Phrase(n.nomecliente));
            table.AddCell(new Phrase(n.custo.ToString()));
            table.AddCell(new Phrase(n.morada));
            table.AddCell(new Phrase(n.relatorio));

        }
        doc.Add(table);
        EmpCorrente_documento = caminho;
        Process.Start(@caminho);
        //é aberto o documento pdf apos a escrita dos dados.

    }
    catch (DocumentException dex)
    {
        throw (dex);
    }
    catch (IOException ioex)
    {
        throw (ioex);
    }
    finally
    {
        doc.Close(); //fecha o documento
    }
}

public void PdfComTodos() //cria um pdf na pasta documentos com o nome todos, que contem
todos os empregados e os seus registos
{
    reset2();
    int pagenumber = 1;
    Document doc = new Document(PageSize.A4, 15f, 15f, 15f, 15f);
    try
    {
        string caminho = Environment.CurrentDirectory;
        caminho = caminho.Substring(0, caminho.IndexOf("bin")) +
"documentos\\todos.pdf"; //deteta o caminho
        FileStream fs = new FileStream(caminho, FileMode.Create, FileAccess.Write);
        PdfWriter w = PdfWriter.GetInstance(doc, fs);
        doc.Open();
        foreach (empregado emp in ListEmp) //ciclo que corre todos os empregados
        {

```

```

        String fotopath = (emp.fotopath != null) ?
Environment.CurrentDirectory.Substring(0, Environment.CurrentDirectory.IndexOf("bin")) +
"imagens\\" + emp.fotopath :
        Environment.CurrentDirectory.Substring(0,
Environment.CurrentDirectory.IndexOf("bin")) + "imagens\nofoto.png"; //se existir um
photopath do empregado corrente, ele vai buscar esse caminho, se nao vai buscar o caminho do
nofoto(foto atribuida ao empregados sem foto)
        Font titulo1 = FontFactory.GetFont("COURIER", 20f);
        titulo1.SetStyle(Font.BOLD);
        Paragraph p1 = new Paragraph("Registro do Funcionario\n", titulo1);
        p1.Alignment = Element.ALIGN_CENTER;
        doc.Add(p1);
        doc.Add(new Phrase("\n\n"));
        var foto = iTextSharp.text.Image.GetInstance(fotopath); // imagem do
empregado
        foto.Alignment = iTextSharp.text.Image.ALIGN_CENTER;
        foto.ScaleAbsoluteHeight(200);
        foto.ScaleAbsoluteWidth(170);
        doc.Add(foto);
        doc.Add(new Phrase("\n\n\n"));
        Font titulo2 = FontFactory.GetFont("COURIER", 15f);
        Paragraph p2 = new Paragraph("Id: " + emp.idemp.ToString() + " Nome: "
+ emp.nome, titulo2); //id e nome do empregado
        p2.IndentationLeft = 120;
        p2.SetLeading(5.0f, 4.0f);
        doc.Add(p2);
        doc.Add(new Phrase("\n\n"));
        PdfPTable table = new PdfPTable(6);
        table.DefaultCell.BorderColor = BaseColor.BLACK;
        table.DefaultCell.Border = Rectangle.BOX;
        table.DefaultCell.HorizontalAlignment = Element.ALIGN_LEFT;
        PdfPCell cell = new PdfPCell(new Phrase("Pedidos do Funcionario",
        titulo1));
        cell.Colspan = 6;
        cell.HorizontalAlignment = Element.ALIGN_CENTER;
        cell.VerticalAlignment = Element.ALIGN_CENTER;
        table.AddCell(cell);
        table.AddCell(new Phrase("Nº Pedido"));
        table.AddCell(new Phrase("IdEmp"));
        table.AddCell(new Phrase("Nome Cliente"));
        table.AddCell(new Phrase("Custo"));
        table.AddCell(new Phrase("Morada"));
        table.AddCell(new Phrase("Relatorio"));
        foreach (pedido n in emp_pedidos) //ciclo que corre todos os pedidos do
empregado e insere numa tabela
    {
        table.AddCell(new Phrase(n.idpedido.ToString()));
        table.AddCell(new Phrase(n.idemp.ToString()));
        table.AddCell(new Phrase(n.nomecliente));
        table.AddCell(new Phrase(n.custo.ToString()));
        table.AddCell(new Phrase(n.morada));
        table.AddCell(new Phrase(n.relatorio));
    }
    doc.Add(table);
    Paragraph footer = new Paragraph(".....");
    footer.SetLeading(0f, 18f);
    doc.Add(footer);
    LineSeparator line = new LineSeparator(1f, 100f, BaseColor.BLACK,
Element.ALIGN_LEFT, 1);
    doc.Add(line);
    doc.Add(new Paragraph(pagenumber.ToString()));
    doc.NewPage();

```

```

       pagenumber++;
    }
    Process.Start(@caminho);
    //é aberto o documento pdf apos a escrita dos dados.

}
catch (DocumentException dex)
{
    throw (dex);
}
catch (IOException ioex)
{
    throw (ioex);
}
finally
{
    doc.Close();
    //fecha o documento
}

}

#endregion
}
}

```

ViewModelPedidos.cs

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.ComponentModel;
using System.Collections.ObjectModel;
using System.Windows.Data;
using System.Windows.Controls;
using Microsoft.Win32;
using System.Data.Entity;
using System.Runtime.InteropServices;
using Microsoft.Win32.SafeHandles;
using System.Collections.Specialized;
using System.Windows.Media.Imaging;
using System.Windows;
using System.Data.Entity.Infrastructure;
using System.IO;
using System.Windows.Media;
using System.Globalization;

namespace FinalProject
{
    public class ViewModelPedidos : INotifyPropertyChanged
    {
        #region Class
        public istecEntities bd; // criação de um modelo onde contem as informações dos
        empregados

        private ObservableCollection<empregado> _listEmp;

```

```

public ObservableCollection<empregado> ListEmp //construtor do observablecollection
{
    get { return _listEmp; }
    set
    {
        _listEmp = value;
        OnPropertyChanged("ListEmp");
    }
}

public ICollectionView _viewEmp;
public ICollectionView ViewEmp //construtor do icollection
{
    get { return _viewEmp; }
    set
    {
        _viewEmp = value;
        OnPropertyChanged("ViewEmp");
    }
}

public empregado EmpCorrente { get; set; }
#endregion
void teste()
{
    ListEmp = new
ObservableCollection<empregado>(bd.empregados.Include("pedidos").ToList());
}
private void ViewEmp_CurrentChanging(object sender, CurrentChangingEventArgs e)
{
    EmpCorrente = ViewEmp.CurrentItem as empregado;
}

public ViewModelPedidos() //construtor da classe
{
    using (bd = new istecEntities())
    {
        //listar os empregados e coloca na ListView
        ListEmp = new
ObservableCollection<empregado>(bd.empregados.Include("pedidos").ToList());
        ViewEmp = CollectionViewSource.GetDefaultView(ListEmp);
        ViewEmp.CurrentChanging += ViewEmp_CurrentChanging;
        //identifica o empregado selecionado
        EmpCorrente = ViewEmp.CurrentItem as empregado;
    }
}

}
#region PropertyChanged
public event PropertyChangedEventHandler PropertyChanged;
public void OnPropertyChanged(String nome)
{
    if (PropertyChanged != null) PropertyChanged(this, new
PropertyChangedEventArgs(nome));
}
#endregion

#region CRUD
public void inserirpedido(TextBox idempTextBox, TextBox clientTextBox, TextBox
custoTextBox, TextBox moradaTextBox, TextBox relatorioTextBox)//inserir novo pedido
{

```



```

    public void updatepedido(ListView lst) //atualizar o pedido selecionado do empregado
selecionado
    {
        using (bd = new istecEntities())
        {
            var pedido = ((pedido)lst.SelectedItem);
            var este = bd.pedidos.Where(x => x.idemp == pedido.idemp && x.idpedido ==
pedido.idpedido).FirstOrDefault(); //verifica o pedido pelo id e pelo id do empregado e vai
buscar todos os seus dados
            if (este != null)
            {
                //recebe os dados preenchidos

                var p = este.empregado;
                este.idemp = pedido.idemp;
                este.nomecliente = pedido.nomecliente;
                este.custo = Math.Round(pedido.custo, 2);
                este.morada = pedido.morada;
                este.relatorio = pedido.relatorio;
                bd.SaveChanges(); //grava na base de dados

                //atualiza a parte grafica e o pedido corrente é o que foi atualizado
onPropertyChanged("ListEmp");
                ListEmp = new
ObservableCollection<empregado>(bd.empregados.Include("pedidos").ToList());
                ViewEmp = CollectionViewSource.GetDefaultView(ListEmp);
                ViewEmp.CurrentChanging += ViewEmp_CurrentChanging;
                ViewEmp.MoveCurrentTo(p);
                EmpCorrente = ViewEmp.CurrentItem as empregado;
                System.Windows.MessageBox.Show("Atualizado com Sucesso");
            }
            else { MessageBox.Show("Verificar os Campos"); }
        }
    }

    public void apagarpedido(System.Windows.Controls.ListView lst) //apagar o pedido
selecionado do empregado selecionado
    {
        using (bd = new istecEntities())
        {
            var pedido = ((pedido)lst.SelectedItem);
            var este = bd.pedidos.Where(x => x.idemp == pedido.idemp && x.idpedido ==
pedido.idpedido).FirstOrDefault(); //verifica o pedido pelo id e pelo id do empregado e vai
buscar todos os seus dados

            if (este != null)
            {
                try {
                    var p = este.empregado;
                    p.pedidos.Remove(este); //apaga o pedido e todos os seus dados
                    bd.SaveChanges(); //grava na base de dados
                    //atualiza a parte grafica, mantem o empregado corrente e mostra os
seus pedidos
                    onPropertyChanged("ListEmp");
                    ListEmp = new
ObservableCollection<empregado>(bd.empregados.Include("pedidos").ToList());
                    ViewEmp = CollectionViewSource.GetDefaultView(ListEmp);
                    ViewEmp.CurrentChanging += ViewEmp_CurrentChanging;
                    ViewEmp.MoveCurrentTo(p);
                    EmpCorrente = ViewEmp.CurrentItem as empregado;
                    System.Windows.MessageBox.Show("Apagado com Sucesso");
                }
            }
        }
    }

```

```

        catch (Exception erro)
        {
            MessageBox.Show(erro.Message);
        }
    }

}

#endregion
}
}

```

autentica.cs

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.DirectoryServices;

namespace FinalProject
{
    class autentica
    {
        public bool AuthenticateUser(string domainName, string userName, string password)
        {
            bool ret = false;
            try
            {
                DirectoryEntry de = new DirectoryEntry("LDAP://" + domainName, userName,
password);
                DirectorySearcher dsearch = new DirectorySearcher(de);
                SearchResult results = null;
                results = dsearch.FindOne();
                ret = true;
            }
            catch
            {
                ret = false;
            }
            return ret;
        }
    }
}

```

Comandos.cs

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Input;

```

```

namespace FinalProject
{
    public static class Comandos
    {
        public static RoutedUICommand sair = new RoutedUICommand("Sair", "sair",
typeof(Comandos));
        public static RoutedUICommand navegarpag1 = new RoutedUICommand("Navegarpag1",
"navegarpag1", typeof(Comandos));
        public static RoutedUICommand navegarpag2 = new RoutedUICommand("Navegarpag2",
"navegarpag2", typeof(Comandos));
        public static RoutedUICommand navegarpag3 = new RoutedUICommand("Navegarpag3",
"navegarpag3", typeof(Comandos));
        public static RoutedUICommand gofirst = new RoutedUICommand("Go First", "gofirst",
typeof(Comandos));
        public static RoutedUICommand gonext = new RoutedUICommand("Go Next", "gonext",
typeof(Comandos));
        public static RoutedUICommand golast = new RoutedUICommand("Go Last", "golast",
typeof(Comandos));
        public static RoutedUICommand goprevious = new RoutedUICommand("Go Previous",
"goprevious", typeof(Comandos));
        public static RoutedUICommand updateEmp = new RoutedUICommand("updateEmp",
"updateEmp", typeof(Comandos));
        public static RoutedUICommand deleteEmp = new RoutedUICommand("deleteEmp",
"deleteEmp", typeof(Comandos));
        public static RoutedUICommand inserirEmp = new RoutedUICommand("inserirEmp",
"inserirEmp", typeof(Comandos));
        public static RoutedUICommand loadimagem = new RoutedUICommand("loadimagem",
"loadimagem", typeof(Comandos));
        public static RoutedUICommand insertimagem = new RoutedUICommand("insertimagem",
"insertimagem", typeof(Comandos));
        public static RoutedUICommand deletepedido = new RoutedUICommand("deletepedido",
"deletepedido", typeof(Comandos));
        public static RoutedUICommand inserirpedido = new RoutedUICommand("inserirpedido",
"inserirpedido", typeof(Comandos));
        public static RoutedUICommand updatepedido = new RoutedUICommand("updatepedido",
"updatepedido", typeof(Comandos));
        public static RoutedUICommand guardaPdf = new RoutedUICommand("guardaPdf",
"guardaPdf", typeof(Comandos));
        public static RoutedUICommand PdfComTodos = new RoutedUICommand("PdfComTodos",
"PdfComTodos", typeof(Comandos));
        public static RoutedUICommand ordenarid = new RoutedUICommand("ordenarid",
"ordenarid", typeof(Comandos));
        public static RoutedUICommand ordenarnome = new RoutedUICommand("ordenarnome",
"ordenarnome", typeof(Comandos));
    }
}

```

Login.xaml

```

<Window x:Class="FinalProject.Login"
    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:FinalProject"
    mc:Ignorable="d"
    Title="Projeto Final" Height="275" Width="420" ResizeMode="NoResize">
<Grid Background="LightSteelBlue">

```

```

        <TextBox Name="txtlog" Width="150" Height="20" HorizontalAlignment="Right"
Margin="0,37,75,125"></TextBox>
        <PasswordBox Name="boxpw" Width="150" Height="20" PasswordChar="*"
HorizontalAlignment="Right" Margin="0,85,75,85"></PasswordBox>
        <Label Width="80" Height="30" Content="Username" HorizontalAlignment="Left"
Margin="60,37,0,125" ></Label>
        <Label Width="80" Height="30" Content="Password" HorizontalAlignment="Left"
Margin="60,85,0,85" ></Label>
        <Button Width="50" Height="25" Content="Validar" Margin="265,125,75,30"
Click="Button_Click"></Button>
    </Grid>
</Window>

```

Login.xaml.cs

```

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 FinalProject
{
    /// <summary>
    /// Interaction logic for Login.xaml
    /// </summary>
    public partial class Login : Window
    {
        String usr;
        String pw;
        String nmdominio = "a053.istec";
        MainWindow wind = new MainWindow();

        public Login()
        {
            InitializeComponent();
        }

        private void Button_Click(object sender, RoutedEventArgs e)
        {
            usr = txtlog.Text;
            pw = boxpw.Password.ToString();
            autentica ad = new autentica();

            if (ad.AuthenticateUser(nmdominio, usr, pw))
            {
                this.Close();
                MessageBox.Show("Login Valido");
                wind.ShowDialog();
                //DialogResult = true;
            }
        }
    }
}

```

```

        else
            MessageBox.Show("Login Errado, tente de novo");
            boxpw.Clear();
        }
    }
}

```

MainWindow.xaml

```

<Window x:Class="FinalProject.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:FinalProject"
    mc:Ignorable="d"
    Title="Projeto Final" Height="600" Width="800">
<Window.Resources>
    <local:ViewModel x:Key="model"></local:ViewModel>
</Window.Resources>
<Grid Background="LightSteelBlue">
    <Grid.RowDefinitions>
        <RowDefinition Height="auto"/>
        <RowDefinition Height="*"/>
    </Grid.RowDefinitions>
    <DockPanel Grid.Row="0">

        <Menu DockPanel.Dock="Top" >
            <MenuItem Header="Exit" Command="local:Comandos.sair"></MenuItem>
            <MenuItem Header="Navegação">
                <MenuItem Name="item1"
                    Command="local:Comandos.navegarpag1"
                    Header="Página 1" ></MenuItem>
                <MenuItem
                    Name="item2"
                    Command="local:Comandos.navegarpag2"
                    Header="Página 2"></MenuItem>
                <MenuItem
                    Name="item3"
                    Command="local:Comandos.navegarpag3"
                    Header="Página 3">
                    </MenuItem>
                </MenuItem>
            </Menu>
        </DockPanel>
        <Frame Grid.Row="2" Name="frame" Source="Page1.xaml"
NavigationUIVisibility="Visible"></Frame>
    </Grid>
</Window>

```

MainWindow.xaml.cs

```

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.Navigation;
using System.Windows.Shapes;

namespace FinalProject
{
    /// <summary>
    /// Interaction logic for MainWindow.xaml
    /// </summary>
    public partial class MainWindow : Window
    {
        public ViewModel vm { get; set; }
        public MainWindow()
        {
            InitializeComponent();
            vm = (ViewModel)Resources["model"];
            vm.w = this;
            this.DataContext = vm;
            #region Comandos
            CommandBindings.Add(new CommandBinding(
                Comandos.sair,
                (sender, e) => vm.sair(),
                (sender, e) => e.CanExecute = true
            ));
            CommandBindings.Add(new CommandBinding(
                Comandos.navegarpag1,
                (sender, e) => vm.navegar("Page1"),
                (sender, e) => e.CanExecute = true
            ));
            CommandBindings.Add(new CommandBinding(
                Comandos.navegarpag2,
                (sender, e) => vm.navegar("Page2"),
                (sender, e) => e.CanExecute = true
            ));
            CommandBindings.Add(new CommandBinding(
                Comandos.navegarpag3,
                (sender, e) => vm.navegar("Page3"),
                (sender, e) => e.CanExecute = true
            ));
            #endregion
        }
    }
}

```

Page1.xaml

```

<Page x:Class="FinalProject.Page1"
      xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
      xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
      xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
      xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
      xmlns:local="clr-namespace:FinalProject"
      mc:Ignorable="d"
      d:DesignHeight="600" d:DesignWidth="800"
      Title="Page1">
    <Page.Resources>
        <local:conversor x:Key="xpto"></local:conversor>
    </Page.Resources>

```

```

<Grid Margin="10" >
    <Grid.RowDefinitions>
        <RowDefinition Height="Auto" />
        <RowDefinition Height="Auto" />
        <RowDefinition Height="Auto" />
        <RowDefinition Height="*" />
        <RowDefinition Height="*" />
    </Grid.RowDefinitions>
    <!--Filtering toolbar-->
    <Expander Header="Agrupar/Procurar" Grid.Row="2">
        <DockPanel >
            <TextBlock Margin="5,0,5,0" Text="Procurar"></TextBlock>
            <TextBox Name="txtproc" Width="200" HorizontalAlignment="Left" Text="{Binding SearchText, UpdateSourceTrigger=PropertyChanged}"></TextBox>
            <CheckBox Content="Agrupar Por Função" Margin="10,0,0,0" IsChecked="{Binding GroupByClube}" />
        </DockPanel>
    </Expander>
    <Border Margin="0,20,20,0" Grid.Row="3" HorizontalAlignment="Right" Height="170" Background="White" Width="150" BorderBrush="Gray" BorderThickness="4">
        <Image Name="img" VerticalAlignment="Center" Source="{Binding EmpCorrente.fotopath, Converter={StaticResource xpto}}" ></Image>
    </Border>
    <ListView Name="lst" Grid.Row="3" BorderBrush="Silver" BorderThickness="5" ItemsSource="{Binding ListEmp}" Margin="15,5,200,0" IsSynchronizedWithCurrentItem="True">
        <ListView.GroupStyle>
            </ListView.GroupStyle>
        <ListView.View >
            <GridView>
                <GridViewColumn Header="ID" DisplayMemberBinding="{Binding idemp}" Width="35"/>
                <GridViewColumn Header="Nome" DisplayMemberBinding="{Binding nome}" Width="315"/>
                <GridViewColumn Header="Função" DisplayMemberBinding="{Binding funcao}" Width="85" />
                <GridViewColumn Header="Nº Telefone" DisplayMemberBinding="{Binding telemovel}" Width="90" />
            </GridView>
        </ListView.View>
    </ListView>

    <StackPanel Grid.Row="4" Margin="7">
        <!--Controls to move the selection-->
        <Border CornerRadius="3" BorderBrush="Gray" BorderThickness="1" HorizontalAlignment="Right" Margin="0,0,59,0">
            <Button Content="Load Foto" HorizontalAlignment="Right" Command="local:Comandos.loadimagem" VerticalAlignment="Top"></Button>
        </Border>
        <Button Content="Ordenar ID" Margin="10,0,0,0" HorizontalAlignment="Left" Command="local:Comandos.ordenarid" VerticalAlignment="Top"></Button>
        <Button HorizontalAlignment="Left" Content="Ordenar Nome" VerticalAlignment="Top" Command="local:Comandos.ordenarnome" Margin="90,-20,0,0" ></Button>
        <!--<TextBox Name="txtproc" Margin="190,-18,0,0" Width="200" HorizontalAlignment="Left" Text="{Binding SearchText, UpdateSourceTrigger=PropertyChanged}"></TextBox>-->
        <StackPanel Orientation="Horizontal" TextElement.FontWeight="Bold" Grid.ColumnSpan="2" HorizontalAlignment="Center" Margin="0,-25,0,0">
            <Button BorderBrush="Silver" BorderThickness="2" Content="<|>" Margin="5" Command="local:Comandos.gofirst" FontWeight="Bold" />
        </StackPanel>
    </StackPanel>

```

```

        <Button BorderBrush="Silver" BorderThickness="2" Content="<" Margin="5"
Command="local:Comandos.goprevious" FontWeight="Bold" />
        <Button BorderBrush="Silver" BorderThickness="2" Content=">" Margin="5"
Command="local:Comandos.gonext" FontWeight="Bold" />
        <Button BorderBrush="Silver" BorderThickness="2" Content="|>" Margin="5"
Command="local:Comandos.golast" FontWeight="Bold" />
    </StackPanel>

    <Border Margin="10,10,200,0" CornerRadius="5" BorderBrush="Silver"
BorderThickness="5" Padding="5" Background="White">
    <!--Shows the currently selected item-->
    <Grid>
        <Grid.RowDefinitions>
            <RowDefinition/>
            <RowDefinition/>
            <RowDefinition/>
            <RowDefinition/>
            <RowDefinition/>
            <RowDefinition/>
        </Grid.RowDefinitions>
        <Grid.ColumnDefinitions>
            <ColumnDefinition Width="100"/>
            <ColumnDefinition />
        </Grid.ColumnDefinitions>
        <TextBlock Text="Funcionário Selecionado" FontWeight="Bold"
Grid.ColumnSpan="2" Grid.Row="0"/>

        <TextBlock Text="ID" Grid.Row="1" Grid.Column="0"
HorizontalAlignment="Right" Margin="0,0,15,0" />
        <TextBlock Name="txtid" Text="{Binding EmpCorrente.idemp, Mode=TwoWay}"
Grid.Row="1" Grid.Column="1"/>

        <TextBlock Text="Nome" Grid.Row="2" Grid.Column="0"
HorizontalAlignment="Right" Margin="0,0,15,0" />
        <TextBox Name="txtname" Text="{Binding EmpCorrente.nome, Mode=TwoWay}"
Grid.Column="1" Grid.Row="2"/>

        <TextBlock Text="Função" Margin="0,0,15,0" Grid.Row="3" Grid.Column="0"
HorizontalAlignment="Right" />
        <TextBox Name="txtfunc" Text="{Binding EmpCorrente.funcao, Mode=TwoWay}"
Grid.Column="1" Grid.Row="3"/>

        <TextBlock Text="Nº Telefone" Margin="0,0,15,0" Grid.Row="4"
Grid.Column="0" HorizontalAlignment="Right" />
        <TextBox Name="txttele" Text="{Binding EmpCorrente.telemovel,
Mode=TwoWay}" Grid.Column="1" Grid.Row="4"/>

        <!--<Button Content="Load Foto" Grid.Column="3" Grid.Row="5"></Button>-->
        <Border BorderThickness="1" HorizontalAlignment="Right"
BorderBrush="Silver" Grid.Column="1" Grid.Row="5" Margin="0,5,5,1" Padding="5"
CornerRadius="5">
        <DockPanel>
            <Border CornerRadius="3" BorderBrush="Gray" BorderThickness="1"
HorizontalAlignment="Right" Margin="5,5,5,5">
                <Button Content="Update" DockPanel.Dock="Right"
Command="local:Comandos.updateEmp" HorizontalAlignment="Right" />
            </Border>

```

```

        <Border CornerRadius="3" BorderBrush="Gray" BorderThickness="1"
HorizontalAlignment="Right" Margin="5,5,5,5">
            <Button Content="Delete" DockPanel.Dock="Right"
Command="local:Comandos.deleteEmp" HorizontalAlignment="Right"></Button>
        </Border>
        <Border CornerRadius="3" BorderBrush="Gray" BorderThickness="1"
HorizontalAlignment="Right" Margin="5,5,5,5">
            <Button Content="Insert" DockPanel.Dock="Right"
Command="local:Comandos.navegarpag2" HorizontalAlignment="Right" ></Button>
        </Border>
    </DockPanel>
</Border>
</Grid>
</Border>
<Border CornerRadius="3" BorderBrush="Gray" BorderThickness="1"
HorizontalAlignment="Right" Margin="0,-120,60,120">
    <Button Content="Download" Command="local:Comandos.guardaPdf"
HorizontalAlignment="Right" Grid.Row="4" ></Button>
</Border>
<Border CornerRadius="3" BorderBrush="Gray" BorderThickness="1"
HorizontalAlignment="Right" Margin="0,-90,40,50">
    <Button Content="Download Todos" Command="local:Comandos.PdfComTodos"
HorizontalAlignment="Right" Grid.Row="4" ></Button>
</Border>
<!--<Image Name="logo" Source="imagens\logo.png" HorizontalAlignment="right"
Height="129" Margin="0,-150,25,0" Width="130" Grid.Row="4" />-->
</StackPanel>

</Grid>

</Page>

```

Page1.xaml.cs

```

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.Navigation;
using System.Windows.Shapes;
using System.Globalization;
using System.IO;
using System.Web;

namespace FinalProject
{
    /// <summary>
    /// Interaction logic for Page1.xaml
    /// </summary>
    public partial class Page1 : Page
    {

```

```

public ViewModel vm { get; set; }
public Page1()
{
    InitializeComponent();
    MainWindow w = Application.Current.MainWindow as MainWindow;
    vm = w.vm;
    vm.onPropertyChanged("ListEmp");
    this.DataContext = vm;

    #region Comando
    CommandBindings.Add(new CommandBinding(
        Comandos.gofirst,
        (sender, e) => vm.gofirst(),
        (sender, e) => e.CanExecute = vm.canGoFirst()
    ));
    CommandBindings.Add(new CommandBinding(
        Comandos.gonext,
        (sender, e) => vm.goNext(),
        (sender, e) => e.CanExecute = vm.canGoLast()
    ));
    CommandBindings.Add(new CommandBinding(
        Comandos.goprevious,
        (sender, e) => vm.goPrevious(),
        (sender, e) => e.CanExecute = vm.canGoFirst()
    ));
    CommandBindings.Add(new CommandBinding(
        Comandos.golast,
        (sender, e) => vm.goLast(),
        (sender, e) => e.CanExecute = vm.canGoLast()
    ));
    CommandBindings.Add(new CommandBinding(
        Comandos.updateEmp,
        (sender, e) => vm.updateEmp(vm.EmpCorrente),
        (sender, e) => e.CanExecute = true
    ));

    CommandBindings.Add(new CommandBinding(
        Comandos.deleteEmp,
        (sender, e) => vm.deleteEmp(vm.EmpCorrente),
        (sender, e) => e.CanExecute = true
    ));
    CommandBindings.Add(new CommandBinding(
        Comandos.loadimagem,
        (sender, e) => vm.loadimagem(img, vm.EmpCorrente),
        (sender, e) => e.CanExecute = true
    ));

    CommandBindings.Add(new CommandBinding(
        Comandos.guardaPdf,
        (sender, e) => vm.guardaPdf(vm.EmpCorrente.idemp),
        (sender, e) => e.CanExecute = true
    ));
    CommandBindings.Add(new CommandBinding(
        Comandos.PdfComTodos,
        (sender, e) => vm.PdfComTodos(),
        (sender, e) => e.CanExecute = true
    ));
    CommandBindings.Add(new CommandBinding(
        Comandos.ordenarid,
        (sender, e) => vm.ordenarid(),
        (sender, e) => e.CanExecute = true
    );
}

```

```

        );
        CommandBindings.Add(new CommandBinding(
            Comandos.ordenarnome,
            (sender, e) => vm.ordenarnome(),
            (sender, e) => e.CanExecute = true
        ));
    #endregion
}
}

public class conversor : IValueConverter
{
    public object Convert(object value, Type targetType, object parameter, CultureInfo culture)
    {
        string fich = Environment.CurrentDirectory;
        fich = fich.Substring(0, fich.IndexOf("bin")) + "imagenes\\";
        // stringnofoto = fich += "nofoto.png";
        BitmapImage foto = null;
        if (value != null && !String.IsNullOrEmpty(value.ToString())) fich += value.ToString();
        else fich += "nofoto.png";
        if (File.Exists(fich))
        {
            foto = new BitmapImage();
            foto.BeginInit();
            foto.UriSource = new Uri(fich, UriKind.Absolute);
            foto.EndInit();
            return foto;
        }
        else
        {
            foto = new BitmapImage();
            foto.BeginInit();
            foto.UriSource = new Uri(@"imagens\nofoto.png", UriKind.Relative);
            foto.EndInit();
            return foto;
        }
    }

    public object ConvertBack(object value, Type targetType, object parameter, CultureInfo culture)
    {
        throw new NotImplementedException();
    }
}
}

```

Page2.xaml

```

<Page x:Class="FinalProject.Page2" VerticalAlignment="Center" HorizontalAlignment="Center"
      xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
      xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
      xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
      xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
      xmlns:local="clr-namespace:FinalProject"
      mc:Ignorable="d"

```

```

d:DesignHeight="600" d:DesignWidth="800"
Title="Page2">

<Grid>
    <Grid x:Name="grid1" HorizontalAlignment="Center" VerticalAlignment="Center" >

        <Grid.ColumnDefinitions>
            <ColumnDefinition Width="Auto"/>
            <ColumnDefinition Width="26" />
            <ColumnDefinition Width="358"/>
            <ColumnDefinition Width="Auto"/>
            <ColumnDefinition Width="39*"/>
            <ColumnDefinition Width="206*"/>
        </Grid.ColumnDefinitions>
        <Grid.RowDefinitions>
            <RowDefinition Height="Auto"/>
            <RowDefinition Height="Auto"/>
            <RowDefinition Height="Auto"/>
            <RowDefinition Height="Auto"/>
            <RowDefinition Height="Auto"/>
            <RowDefinition Height="Auto"/>
            <RowDefinition Height="Auto"/>
        </Grid.RowDefinitions>
        <Label Content="Inserir Novo Registo" HorizontalAlignment="Center" FontSize="26"
Grid.Row="0" Grid.Column="2" Margin="0,-100,0,0"></Label>

        <Label Content="*Id:" Grid.Column="0" HorizontalAlignment="Right"
Margin="0,12,2,12" Grid.Row="1" VerticalAlignment="Center" />
        <TextBox x:Name="idTextBox" Grid.Column="1" Height="23" Margin="4,14,1,14"
Grid.Row="1" Grid.ColumnSpan="2" />

        <Label Content="*Nome:" Grid.Column="0" HorizontalAlignment="Right"
Margin="0,12,2,12" Grid.Row="2" VerticalAlignment="Center"/>
        <TextBox x:Name="nomeTextBox" Grid.Column="1" Height="23" Margin="4,14,1,14"
Grid.Row="2" Grid.ColumnSpan="2" />

        <Label Content="*Função:" Grid.Column="0" HorizontalAlignment="Right"
Margin="0,12,2,12" Grid.Row="3" VerticalAlignment="Center"/>
        <TextBox x:Name="teleTxt" Grid.Column="1" Height="23" Margin="4,14,1,14"
Grid.Row="4" Grid.ColumnSpan="2" />

        <Label Content="*Nº Telemovel:" Grid.Column="0" HorizontalAlignment="Right"
Margin="0,12,2,12" Grid.Row="4" VerticalAlignment="Center"/>

        <Image x:Name="fotoImage" Grid.Column="4" Width="150" Grid.RowSpan="5"
HorizontalAlignment="Left" Margin="40,10,10,10" Height="170" Grid.ColumnSpan="2" />

        <Button Grid.Column="2" HorizontalAlignment="Right" Grid.Row="5"
Command="local:Comandos.inserirEmp" Height="23" Content="Inserir" Width="70"/>

        <Button Grid.Column="5" Grid.Row="5" Command="local:Comandos.insertimagem"
Margin="0,0,20,0" Width="85" Content="Carregar Foto" HorizontalAlignment="Center" />
        <ComboBox Name="cmb" Grid.Column="1" Margin="4,14,1,10" Grid.Row="3" Height="26"
VerticalAlignment="Center" Grid.ColumnSpan="2">
            <ComboBox.ItemTemplate>
                <DataTemplate>
                    <TextBlock Text="{Binding funcao}"></TextBlock>
                </DataTemplate>
            </ComboBox.ItemTemplate>
        </ComboBox>
    </Grid>

```

```

        <Label Content="* Campos de preenchimento obrigatorio" Grid.Column="2"
Grid.Row="6" HorizontalAlignment="Left" VerticalAlignment="Bottom" FontSize="16" Margin="-
90,0,0,0"></Label>

    </Grid>
</Grid>
</Page>
```

Page2.xaml.cs

```

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.Navigation;
using System.Windows.Shapes;

namespace FinalProject
{
    /// <summary>
    /// Interaction logic for Page2.xaml
    /// </summary>
    public partial class Page2 : Page
    {
        public MainWindow mainwindow { get; set; }
        ViewModel vm { get; set; }
        public Page2()
        {
            InitializeComponent();
            mainwindow = (MainWindow)Application.Current.MainWindow as MainWindow;
            vm = mainwindow.vm;
            this.DataContext = vm;
            this.cmb.ItemsSource = vm.Funcao;
            CommandBindings.Add(new CommandBinding(
                Comandos.insertimagem,
                (sender, e) => vm.insertimagem(fotoImage),
                (sender, e) => e.CanExecute = true
            ));
            CommandBindings.Add(new CommandBinding(
                Comandos.inserirEmp,
                (sender, e) => vm.inserirEmp(idTextBox, nomeTextBox, cmb, fotoImage, teleTxt),
                (sender, e) => e.CanExecute = true
            ));
        }
    }
}
```

}

Page3.xaml

```
<Page x:Class="FinalProject.Page3"
    xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
    xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
    xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
    xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
    xmlns:local="clr-namespace:FinalProject"
    mc:Ignorable="d"
    d:DesignHeight="600" d:DesignWidth="800"
    Title="Page3">
<Page.Resources>
    <local:ViewModelPedidos x:Key="bd"></local:ViewModelPedidos>
    <local:converte x:Key="top"></local:converte>
    <local:DecimalConverter x:Key="decimalconverter"></local:DecimalConverter>
</Page.Resources>
<Grid DataContext="{Binding Source={StaticResource bd}}>
    <Grid.RowDefinitions>
        <RowDefinition Height="175*"/>
        <RowDefinition Height="165*"/>
        <RowDefinition Height="150*"/>
    </Grid.RowDefinitions>
    <Border Grid.Row="0" HorizontalAlignment="Right" Height="170" Margin="0,10,70,0"
    Width="150" BorderBrush="Gray" BorderThickness="4" Background="White">
        <Image Name="img" Source="{Binding SelectedItem.fotopath, ElementName=lst,
    Converter={StaticResource top}}></Image>
    </Border>
    <ListView Grid.Row="0" ItemsSource="{Binding ListEmp}" Name="lst"
    Margin="0,0,300,0" IsSynchronizedWithCurrentItem="True">
        <ListView.View>
            <GridView>
                <GridViewColumn Header="ID Emp" DisplayMemberBinding="{Binding idemp}"
    Width="50"/>
                <GridViewColumn Header="Nome" DisplayMemberBinding="{Binding nome}"
    Width="170"></GridViewColumn>
                <GridViewColumn Header="Função" DisplayMemberBinding="{Binding funcao}"
    Width="100" />
            </GridView>
        </ListView.View>
    </ListView>
    <ListView Margin="0,15,0,0" Grid.Row="1" Name="lstntas"
    IsSynchronizedWithCurrentItem="True" ItemsSource="{Binding SelectedItem_pedidos,
    ElementName=lst}">
        <ListView.View>
            <GridView>
                <GridViewColumn Header="Nº Pedido" Width="65"
    DisplayMemberBinding="{Binding idpedido}"/>
                <GridViewColumn Header="Id Emp" Width="50" >
                    <GridViewColumn.CellTemplate>
                        <DataTemplate>
                            <TextBox TextAlignment="Left" Text="{Binding idemp}"
    Width="50" HorizontalAlignment="Left" IsEnabled="False" ></TextBox>
                        </DataTemplate>
                    </GridViewColumn.CellTemplate>
                </GridViewColumn>
            </GridView>
        </ListView.View>
    </ListView>
</Grid>
```

```

        </GridViewColumn>
        <GridViewColumn Header="Cliente" Width="80" >
            <GridViewColumn.CellTemplate>
                <DataTemplate>
                    <TextBox HorizontalAlignment="Left" Text="{Binding nomecliente}" TextAlignment="Left" Width="175" ></TextBox>
                </DataTemplate>
            </GridViewColumn.CellTemplate>

        </GridViewColumn>
        <GridViewColumn Header="Custo" Width="60" >
            <GridViewColumn.CellTemplate>
                <DataTemplate>
                    <TextBox TextAlignment="Left" Text="{Binding custo, Converter={StaticResource decimalconverter}}" HorizontalAlignment="Left" Width="75" ></TextBox>
                </DataTemplate>
            </GridViewColumn.CellTemplate>

        </GridViewColumn>
        <GridViewColumn Header="Morada" Width="150" >
            <GridViewColumn.CellTemplate>
                <DataTemplate>
                    <TextBox HorizontalAlignment="Left" Text="{Binding morada}" TextAlignment="Left" Width="400" ></TextBox>
                </DataTemplate>
            </GridViewColumn.CellTemplate>

        </GridViewColumn>
        <GridViewColumn Header="Relatorio" Width="225" >
            <GridViewColumn.CellTemplate>
                <DataTemplate>
                    <TextBox TextAlignment="Left" Text="{Binding relatorio}" HorizontalAlignment="Left" Width="400" ></TextBox>
                </DataTemplate>
            </GridViewColumn.CellTemplate>

        </GridViewColumn>
        <GridViewColumn Header="Atualizar" Width="70">
            <GridViewColumn.CellTemplate>
                <DataTemplate>
                    <StackPanel Margin="6,2,6,2">
                        <Button Content="Update" Command="local:Comandos.updatepedido" />
                    </StackPanel>
                </DataTemplate>
            </GridViewColumn.CellTemplate>

        </GridViewColumn>

        <GridViewColumn Header="Apagar" Width="70">
            <GridViewColumn.CellTemplate>
                <DataTemplate>
                    <StackPanel Margin="6,2,6,2">
                        <Button Content="Delete" Command="local:Comandos.deletepedido"/>
                    </StackPanel>
                </DataTemplate>
            </GridViewColumn.CellTemplate>

```

```

        </GridViewColumn>

    </GridView>
</ListView.View>

</ListView>

<Grid x:Name="grid1" HorizontalAlignment="Center" Margin="30,30,0,0" Grid.Row="3"
VerticalAlignment="Top">
    <Grid.ColumnDefinitions>
        <ColumnDefinition Width="Auto"/>
        <ColumnDefinition Width="Auto"/>
        <ColumnDefinition Width="Auto"/>
        <ColumnDefinition Width="Auto"/>
    </Grid.ColumnDefinitions>
    <Grid.RowDefinitions>
        <RowDefinition Height="Auto"/>
        <RowDefinition Height="Auto"/>
        <RowDefinition Height="Auto"/>
        <RowDefinition Height="Auto"/>
    </Grid.RowDefinitions>
    <Label Content="Id Empregado:" Grid.Column="0" HorizontalAlignment="right"
Margin="3" Grid.Row="0" VerticalAlignment="Center"/>
        <TextBox x:Name="idempTextBox" Text="{Binding SelectedItem.idemp,
ElementName=lst}" Grid.Column="1" HorizontalAlignment="Right" Margin="3" Grid.Row="0"
Width="220"/>

    <Label Content="Cliente:" Grid.Column="0" HorizontalAlignment="Right" Margin="3"
Grid.Row="1" VerticalAlignment="Center"/>
        <TextBox x:Name="clientTextBox" Grid.Column="1" HorizontalAlignment="Right"
Margin="3" Grid.Row="1" Width="220"/>
    <Label Content="Custo:" Grid.Column="0" HorizontalAlignment="Right" Margin="3"
Grid.Row="2" VerticalAlignment="Center"/>
        <TextBox x:Name="custoTextBox" Grid.Column="1" HorizontalAlignment="Right"
Margin="3" Grid.Row="2" Height="27" Width="220"/>

    <Label Content="Morada:" Grid.Column="2" HorizontalAlignment="Right" Margin="3"
Grid.Row="0" VerticalAlignment="Center" />
        <TextBox x:Name="moradaTextBox" Grid.Column="3" HorizontalAlignment="Right"
Margin="3" Grid.Row="0" Width="220"/>

    <Label Content="Relatorio:" Grid.Column="2" HorizontalAlignment="Right"
Margin="3" Grid.Row="1" VerticalAlignment="Center" />
        <TextBox x:Name="relatorioTextBox" Grid.Column="3" HorizontalAlignment="Right"
Margin="3" Grid.Row="1" Width="220"/>

    <Button Content="Inserir Pedido" Grid.Row="2"
Command="local:Comandos.inserirpedido" HorizontalAlignment="Right" Grid.Column="4"
Width="100" Margin="10"></Button>

</Grid>

</Grid>
</Page>

```

Page3.xaml.cs

```
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.Navigation;
using System.Windows.Shapes;
using System.Globalization;
using System.IO;
using System.Web;
using System.ComponentModel;
using System.Collections.ObjectModel;
```



```
namespace FinalProject
{
    /// <summary>
    /// Interaction logic for Page3.xaml
    /// </summary>
    public partial class Page3 : Page
    {

        public ViewModelPedidos bd { get; set; }
        public Page3()
        {
            InitializeComponent();
            bd = (ViewModelPedidos)Resources["bd"];

            CommandBindings.Add(
                new CommandBinding(
                    Comandos.inserirpedido,
                    (sender, e) => bd.inserirpedido(idempTextBox, clientTextBox, custoTextBox,
relatorioTextBox, moradaTextBox),
                    (sender, e) => e.CanExecute = true
                ));
            CommandBindings.Add(
                new CommandBinding(
                    Comandos.updatepedido,
                    (sender, e) => bd.updatepedido(lstntas),
                    (sender, e) => e.CanExecute = true
                ));
            CommandBindings.Add(
                new CommandBinding(
                    Comandos.deletepedido,
                    (sender, e) => bd.apagarpedido(lstntas),
                    (sender, e) => e.CanExecute = true
                ));
        }
    }
}
```

```

#region Conversor
public class converte : IValueConverter // este
{
    public object Convert(object value, Type targetType, object parameter, CultureInfo culture)
    {
        string fich = Environment.CurrentDirectory;
        fich = fich.Substring(0, fich.IndexOf("bin")) + "imagens\\";
        // stringnofoto = fich += "nofoto.png";
        BitmapImage foto = null;
        if (value != null && !String.IsNullOrEmpty(value.ToString())) fich += value.ToString();
        else fich += "nofoto.png";
        if (File.Exists(fich))
        {
            foto = new BitmapImage();
            foto.BeginInit();
            foto.UriSource = new Uri(fich, UriKind.Absolute);
            foto.EndInit();
            return foto;
        }
        else
        {
            foto = new BitmapImage();
            foto.BeginInit();
            foto.UriSource = new Uri(@"imagens\nofoto.png", UriKind.Relative);
            foto.EndInit();
            return foto;
        }
    }

    public object ConvertBack(object value, Type targetType, object parameter, CultureInfo culture)
    {
        throw new NotImplementedException();
    }
}
#endregion
#region Conversor2
public class DecimalConverter : IValueConverter
{
    public object Convert(object value, Type targetType, object parameter, CultureInfo culture)
    {
        return value.ToString().Replace(",", ".");
    }

    public object ConvertBack(object value, Type targetType, object parameter, CultureInfo culture)
    {
        return value.ToString().Replace(".", ",");
    }
}
#endregion
}

```