

# Biblioteca Virtual

Lisboa 2015



---

Projeto Global

---

Licenciatura Informática

---

Ano letivo 2014/2015

Coordenador:

Prof. Doutor Pedro Brandão

Realizado por:

Carlos Sota – N.º1895



## **DEDICATÓRIA**

Este trabalho é dedicado à minha família, especialmente aos meus pais, irmão e namorada.

## **AGRADECIMENTOS**

Começo por agradecer à minha família por todo o apoio e incentivo transmitido que permitiu o meu ingresso no ensino superior.

Agradeço também à minha namorada por todo o apoio e força que sempre transmitiu, sem o qual não teria atingido este objetivo.

Por fim agradeço aos meus amigos pelo interesse e apoio demonstrado no meu progresso durante a licenciatura.

A todos eles, o meu muito obrigado.

## RESUMO

O presente projeto apresenta como principal objetivo fomentar a atividade de investigação fundamental e aplicada para contribuir, de forma criadora, para a aplicação e desenvolvimento dos conhecimentos adquiridos durante a licenciatura e na pós-graduação, numa perspetiva de consistência científica.

Proponho portanto no presente projeto o desenvolvimento de uma aplicação para uma biblioteca sendo que o acesso a esta aplicação pelos utilizadores é realizado através de uma máquina virtual, de forma a rentabilizar os equipamentos existentes e maximizando por sua vez o retorno de investimento.

As atividades deste projeto, iniciam-se assim, com uma revisão da literatura da qual resulta uma abordagem à evolução da linguagem SQL, a exposição das características da *Cloud Computing*, o conceito de virtualização, quais os tipos de *hypervisor* assim como a diferença entre estes e por último uma abordagem sobre os componentes do ADO.NET.

Posteriormente, segue-se uma abordagem ao longo da evolução do desenvolvimento da aplicação desde a conceção da base de dados passando pelo desenvolvimento de cada janela aplicacional e terminando com uma aplicação em contexto real com recurso às tecnologias de virtualização.

**Palavras-Chave:** SQL, SGBD, ADO.NET, Cloud Computing, Virtualização

## **ABSTRACT**

This project's main objective is to promote fundamental and applied research activity to contribute, in a creative way, to the implementation and development of the knowledge acquired during the degree in a scientific perspective.

Therefore I propose the development of an application for a library whereas the user's application access is done through a virtual machine in order to maximize the existing equipment and the ROI (return of investment).

The activities for this project began with a literature review which shows an approach to the evolution of SQL language, Cloud Computing features, virtualization concepts, existing types of hypervisors, its differences and ADO.NET components.

Subsequently there is a detailed approach along the application development from the database conception and through each application window development. After the development we will see the application deployment in a testing environment with the help of virtualization.

**Key Words:** SQL, SGBD, ADO.NET, Cloud Computing, Virtualization

## SIGLAS E ABREVIATURAS

IDE	<i>Integrated Development Environment</i>
ROI	<i>Return of Investment</i>
SQL	Structured Query Language
ANSI	American National Standards Institute
ISO	International Standards Organization
SAA-SQL	Systems Application Architecture Database Interface
XML	eXtensible Markup Language
TI	Tecnologias de Informação
SaaS	Software as a Service
IaaS	Infrastructure as a Sevice
PaaS	Platform as a Service
ODBC	<i>Open Database Connecivity</i>
OLE DB	<i>Object Linking and Embedding Data Base</i>
PDF	<i>Portable Document Format</i>
SHA	<i>Secure Hash Algorithm</i>
WPF	<i>Windows Presentation Foundation</i>
OVF	<i>Open Virtualization Format</i>
SP1	<i>Service Pack 1</i>

## INDICE

RESUMO.....	IV
ABSTRACT.....	V
1. INTRODUÇÃO.....	13
2. ESTADO DA ARTE .....	14
2.1. SQL e Bases de Dados Relacionais.....	14
2.2. <i>Cloud Computing</i> .....	15
2.2.1. <i>Infrastructure as a Service (IaaS)</i> .....	17
2.2.2. <i>Platform as a Service (PaaS)</i> .....	17
2.2.3. <i>Software as a Service (SaaS)</i> .....	18
2.3. Virtualização.....	19
2.3.1. Tipos de <i>Hypervisor</i> .....	19
2.4. ADO.NET.....	21
2.4.1. Componentes principais do ADO.NET .....	21
2.4.1.1. DataSet .....	22
2.4.1.2. Data Provider.....	22
3. VIRTUAL LIBRARY .....	24
3.1. Pré-Requisitos.....	24
3.2. Estrutura da aplicação .....	25
3.2.1. Modelo de Dados .....	25
3.2.1.1. Tabela TiposPessoa.....	26
3.2.1.2. Tabela Pessoas.....	26
3.2.1.3. Tabela Emails .....	27
3.2.1.4. Tabela Telefones.....	27
3.2.1.5. Tabela Utilizadores .....	28
3.2.1.6. Tabela Editoras.....	28
3.2.1.7. Tabela Autores.....	29
3.2.1.8. Tabela livrosAutores .....	29
3.2.1.9. Tabela Livros.....	30
3.3. Desenvolvimento .....	30

3.3.1.	<i>Class Cripto</i> .....	32
3.3.2.	<i>MainWindow</i> .....	33
3.3.3.	<i>passwordRecover</i> .....	36
3.3.4.	<i>Home</i> .....	38
3.3.5.	<i>Advanced Search</i> .....	45
3.3.6.	<i>Search Results</i> .....	48
3.3.7.	<i>Pessoas</i> .....	52
3.3.8.	<i>Utilizadores</i> .....	63
3.3.9.	<i>Categorias Pessoas</i> .....	68
3.3.10.	<i>Livros</i> .....	71
3.3.11.	<i>Editoras</i> .....	78
3.3.12.	<i>Autores</i> .....	79
3.4.	<b>Virtualização</b> .....	80
3.4.1.	<i>Máquina virtual DC-Biblioteca</i> .....	81
3.4.2.	<i>Máquina virtual SQL</i> .....	81
3.4.3.	<i>Máquina Virtual HyperV</i> .....	81
3.4.4.	<i>Máquina virtual cliente</i> .....	82
	<b>CONCLUSÃO</b> .....	84
	<b>BIBLIOGRAFIA</b> .....	85
	<b>ANEXOS</b> .....	86

## INDICE DE FIGURAS

Figura 1: <i>Hypervisor</i> do Tipo 1. (Vietstack, 2014) .....	20
Figura 2: <i>Hypervisor</i> do Tipo 2. (Vietstack, 2014) .....	20
Figura 3: Diagrama da base de dados .....	25
Figura 4: Tabela TiposPessoa .....	26
Figura 5: Tabela Pessoas .....	26
Figura 6: Tabela Emails .....	27
Figura 7: Tabela Telefones .....	27
Figura 8: Tabela Utilizadores .....	28
Figura 9: Tabela Editoras .....	28
Figura 10: Tabela Autores .....	29
Figura 11: Tabela livrosAutores .....	29
Figura 12: Tabela Livros .....	30
Figura 13: <i>Solution Explorer</i> da aplicação .....	31
Figura 14: Classe Cripto .....	32
Figura 15: Janela <i>MainWindow</i> .....	33
Figura 16: <i>Namespaces</i> da janela <i>MainWindow</i> .....	33
Figura 17: <i>Event handler</i> <i>btnPassLost_Click_1</i> .....	34
Figura 18: <i>Event handler</i> <i>btnLogin_Click_1</i> .....	35
Figura 19: Janela <i>passwordRecover</i> .....	36
Figura 20: Método <i>PasswordRecover</i> .....	36
Figura 21: <i>Event Handler</i> <i>btnClean_Click_1</i> .....	37
Figura 22: <i>Event Handler</i> <i>btnConfirm_Click_1</i> .....	37
Figura 23: Janela <i>Home</i> .....	38
Figura 24: <i>Event handler's</i> da barra de navegação de topo .....	39
Figura 25: Variáveis globais da janela <i>Home</i> .....	40
Figura 26: Método <i>Home</i> .....	40
Figura 27: Método <i>search</i> .....	41
Figura 28: Exemplo do comando SQL retornado pelo método <i>search</i> .....	42
Figura 29: <i>Event handler</i> <i>btnSearch_MouseLeftButtonUp_1</i> .....	43
Figura 30: <i>Event handler</i> <i>lblAdvancedSearch_PreviewMouseLeftButtonDown_1</i> .....	44
Figura 31: Janela <i>Advanced Search</i> .....	45

Figura 32: Método <i>searchArray</i> .....	45
Figura 33: Método <i>search</i> .....	47
Figura 34: Janela <i>searchResults</i> .....	48
Figura 35: Método <i>searchResults</i> .....	48
Figura 36: Método <i>Bind (searchResults)</i> .....	49
Figura 37: <i>Event handler rowDoubleClick</i> .....	50
Figura 38: <i>Event handler openPDF</i> .....	51
Figura 39: <i>Event handler HandleCanExecute</i> .....	52
Figura 40: <i>Event handler Image_MouseLeftButtonDown_1</i> .....	52
Figura 41: Menu Pessoas .....	53
Figura 42: Janela Pessoas .....	53
Figura 43: Método <i>Bind (Pessoas)</i> .....	54
Figura 44: Métodos <i>fillcbPessoa</i> e <i>fillListView</i> .....	55
Figura 45: Método <i>clearValues</i> .....	55
Figura 46: <i>Event Handler btnNext_Click_1</i> .....	56
Figura 47: <i>Event Handler insertDataClick e cancelClick</i> .....	57
Figura 48: <i>Event Handler saveDataClick</i> .....	59
Figura 49: <i>Event Handler deleteClick</i> .....	60
Figura 50: <i>Event Hanlder's addEmail e addContact</i> .....	61
Figura 51: <i>Event Hanlder's removeEmail e removeContact</i> .....	63
Figura 52: Janela Utilizadores.....	63
Figura 53: <i>Event Handler btnPrevious_Click_1</i> .....	64
Figura 54: <i>Event Handler saveDataClick (Utilizadores)</i> .....	66
Figura 55: <i>Event Handler deleteClick (Utilizadores)</i> .....	67
Figura 56: <i>Event Handler saveDataClick (Categorias de Pessoas)</i> .....	69
Figura 57: <i>Event Handler deleteClick (Categorias de Pessoas)</i> .....	70
Figura 58: Janela Livros .....	71
Figura 59: Método <i>bindImage</i> .....	72
Figura 60: <i>Event Handler saveDataClick (Livros)</i> .....	74
Figura 61: <i>Event Handler deleteClick (Livros)</i> .....	75
Figura 62: <i>Event Handler changeCover</i> .....	76
Figura 63: <i>Event Handler changePDF</i> .....	77
Figura 64: Janela Editoras.....	78
Figura 65: Janela Autores .....	79

Figura 66: Propriedades da máquina virtual HyperV .....	82
Figura 67: <i>Virtual Switch Manager</i> da máquina Hyper-V.....	83
Figura 68: Atalho de acesso à aplicação .....	83

## **INDICE DE TABELAS**

Tabela 1: Especificações de <i>hardware</i> do laboratório.....	80
Tabela 2: Especificações de <i>hardware</i> da máquina cliente .....	82

## 1. INTRODUÇÃO

O presente projeto apresenta como principal propósito, fomentar a atividade de investigação fundamental e aplicada para contribuir, de forma criadora, para a aplicação e desenvolvimento dos conhecimentos adquiridos durante a licenciatura, numa perspetiva de consistência científica.

O objetivo principal consiste no desenvolvimento de uma aplicação para uma biblioteca que permita a autenticação de utilizadores, pesquisa de livros, consulta de livros e a gestão e administração da aplicação. Como objetivos específicos definiu-se o desenvolvimento da aplicação em linguagem C# com recurso às tecnologias .NET e a utilização das tecnologias de virtualização para permitir que o acesso à aplicação fosse realizado através de uma máquina virtual garantindo desta forma uma redução nos custos de energia e nos custos dos equipamentos informáticos.

As tecnologias utilizadas para a realização deste projeto seguiram a linha de ferramentas utilizadas ao longo da licenciatura, sendo estas:

- *Microsoft Visual Studio 2012 Ultimate* utilizado como IDE (*Integrated Development Environment*) recorrendo à linguagem de programação C# e às tecnologias .NET;
- *Microsoft SQL Server 2012 SP1* para suporte das bases de dados;
- *VMware Workstation 11* para recriação de um laboratório de teste que simulasse uma rede empresarial.

O projeto encontra-se portanto dividido em dois capítulos principais, sendo o primeiro responsável por uma revisão bibliográfica aprofundada sobre o tema e as tecnologias abordadas. O segundo capítulo encontra-se dividido em três subcapítulos, o primeiro aborda os pré-requisitos, a estrutura e o desenho da base de dados da aplicação, o segundo subcapítulo aborda o desenvolvimento da aplicação com uma visão dos métodos e eventos que constituem a aplicação. O terceiro subcapítulo aborda a realização de um laboratório de teste recorrendo à virtualização e *nested virtualization* para testar a implementação da aplicação num contexto organizacional.

## 2. ESTADO DA ARTE

### 2.1. SQL e Bases de Dados Relacionais

A versão original do SQL chamava-se *Sequel* 2 e foi apresentada por Chamberlin (Chamberlin, et al., 1976) como uma adaptação das linguagens *Square* (Boyce, Chamberlin, Hammer, & King, 1975) e *Sequel* (Chamberlin & Boyce, Sequel: A structured english query language, 1974). Mais tarde o nome foi alterado para *Structured Query Language*, vulgarmente conhecido como SQL. Com o passar dos anos esta linguagem tornou-se a linguagem padrão para todas as bases de dados relacionais.

Em 1986, a ANSI (*American National Standards Institute*) em conjunto com a ISO (*International Standards Organization*) publicaram um conjunto de regras padrão para implementações da linguagem SQL, este padrão ficou conhecido como SQL-86. Um ano mais tarde a IBM, publicou um novo padrão para a linguagem, denominado SAA-SQL (*Systems Application Architecture Database Interface*). No ano de 1989, foi publicado um novo conjunto de regras para o padrão vigente, este novo padrão ficou conhecido como SQL-89. Nas duas décadas seguintes, surgiram outras revisões nos padrões atuais do SQL, a saber, SQL-92, também conhecido como SQL-2, SQL-3 (datado de 1999) e SQL-2003. (Beaulieu, 2009)

A última revisão ao padrão do SQL foi realizada em 2006 e foi denominada como SQL:2006, esta revisão permitiu que fossem adicionadas novas funcionalidades, sendo a principal a incorporação da funcionalidade orientada a objetos, além disso, é destacado ainda a integração do XML (*eXtensible Markup Language*) que originou a linguagem XQuery usada para obter informações de documentos em formato XML. (Beaulieu, 2009)

A linguagem SQL é caracterizada por ser uma linguagem declarativa de pesquisas em bases de dados, e assenta sobre as operações de álgebra relacional, mais concretamente sobre os operadores de seleção, projeção, renomeação, produto cartesiano, união e diferença de conjuntos. (Ramakrishnan & Gehrke).

O SQL possui instruções específicas para consultar, atualizar, inserir e remover dados de uma base de dados relacional, estas instruções são: (W3Schools, 2014)

- SELECT: instrução utilizada para consultar e selecionar informação de uma base de dados relacional;
- INSERT: instrução utilizada para inserir informação numa base de dados relacional;
- UPDATE: instrução utilizada para atualizar informação numa base de dados relacional;
- DELETE: instrução utilizada para eliminar informação numa base de dados relacional.

## **2.2. Cloud Computing**

A computação em Cloud, surge como um novo modelo de computação, emergente na última década.

Segundo Wang, Wang, & Haung (2011), esta é uma nova palavra que surgiu no quarto trimestre de 2007, e que é o sonho de longa data da história da computação, esta possui o potencial de transformar uma grande parte da indústria das tecnologias de informação (TI), fazendo com que o *software* se torne ainda mais atraente como um serviço, e molde a forma como o hardware é desenhado e adquirido. Os programadores podem fazer uso das suas ideias inovadoras para novos serviços de Internet, e não necessitam mais de efetuar enormes investimentos em *hardware* para implementar os seus serviços, ou em custos de mão-de-obra para garantir o seu funcionamento. (Wang, Wang, & Haung, 2011)

Deste modo e ainda segundo os autores supracitados a computação em Cloud refere-se à disponibilização de serviços através da Internet com suporte em *hardware*, e sistemas de software em centros de dados. Os próprios serviços têm sido referidos como *Software* como um Serviço (*Software as a Service* - SaaS). O hardware e software dos centros de dados é o que designamos de Cloud. (Wang, Wang, & Haung, 2011)

Segundo Tim O'Reilly, um outro estudioso do tema, a computação em Cloud é um dos alicerces da nova geração da computação. É um mundo onde as redes são uma plataforma para toda a computação, onde tudo o que pensamos que é um computador é apenas um dispositivo que permite a ligação para um computador maior que estamos a construir. A

computação em Cloud apresenta-se como uma grande forma de pensar no que será possível distribuir ao consumidor final em termos de serviços de computação no futuro. (O'Reilly, 2014)

Existem três novos aspectos relevantes na computação em Cloud e que são parte integrante do seu núcleo. O primeiro aspeto é a ilusão de recursos computacionais ilimitados disponíveis *on demand*<sup>1</sup> para o utilizador da Cloud. Esta propriedade permite aos utilizadores da Cloud colocar em segundo plano as suas preocupações acerca da escalabilidade e aumento de desempenho dos seus sistemas. O segundo aspeto importante possibilita a inexistência de um compromisso em termos de aquisição de novos recursos de *hardware* pelo utilizador da Cloud, esta capacidade/funcionalidade, deriva do primeiro aspeto referido anteriormente. Se a capacidade a nível do *hardware* deixa de ser um problema, as empresas podem aumentar ou reduzir as suas necessidades de poder de computação de acordo com o estado atual de crescimento da empresa ou das necessidades correntes. O terceiro, e último aspeto, é a possibilidade de fornecer um serviço de acordo com a taxa de computacional utilizada. Mais uma vez, estes três aspetos estão interligados, o que nos indica que é possível utilizar os recursos computacionais conforme as nossas necessidades, evitando desta forma grandes investimentos em largas infraestruturas que correm o risco de ficar com uma percentagem de utilização muito baixa. (Armbrust, et al., 2009)

A arquitetura de uma Cloud assenta necessariamente nestes três modelos de abstração:

- *Infrastructure as a Service* (IaaS);
- *Platform as a Service* (PaaS);
- *Software as a Service* (SaaS). (Microsoft, 2013)

---

<sup>1</sup> Recurso disponível por solicitação do utilizador ou empresa.

### **2.2.1. Infrastructure as a Service (IaaS)**

Neste nível estão os componentes fundamentais que possibilitam a esta arquitetura o armazenamento e as capacidades computacionais que serão disponibilizadas aos utilizadores da *Cloud*. O IaaS pode ser considerado como a camada que providencia os recursos físicos, podendo estes depois ser comercializados, através de um modelo de faturação dependente das horas de utilização das instâncias de computação e do volume de dados guardados em armazenamento. A este tipo de utilização dá-se o nome de *Utility Computing*. (Amazon Elastic Compute Cloud (Amazon EC2), 2014)

A Amazon, através da sua solução *Amazon Elastic Compute Cloud* disponibiliza serviços a este nível. A sua solução proporciona aos utilizadores um controlo e acesso total sobre os recursos computacionais que estão a ser utilizados e contratualizados. Isto não significa que o utilizador possui controlo sobre o *Fabric*, mas sim um controlo sobre a máquina virtual ou um conjunto de recursos. A este nível o utilizador deverá estar consciente que o *software* utilizado na máquina virtual é da sua inteira responsabilidade, garantindo consequentemente a sua manutenção. (Amazon Elastic Compute Cloud (Amazon EC2), 2014)

### **2.2.2. Platform as a Service (PaaS)**

Na computação em *Cloud* o segundo nível de abstração é conhecido como *Platform as a Service* (PaaS). Neste nível geralmente já estão instalados e configurados um número de serviços que, quando combinados, oferecem uma plataforma de desenvolvimento, também designada como *framework*, que pode ser utilizada pelo programador da *Cloud* para o desenvolvimento de soluções aplicacionais em SaaS. (Armbrust, et al., 2009)

A imagem de uma máquina virtual com serviços de *software* (um *Web<sup>2</sup> Server*, um ambiente de desenvolvimento, etc.) é o clássico exemplo de uma PaaS, oferecendo desta forma um ambiente *Web* de desenvolvimento para o programador da *Cloud*. Esta configuração é similar às soluções oferecidas pelas empresas de alojamento de conteúdos

---

<sup>2</sup> World Wide Web

*Web*. A diferença reside na forma como as empresas de alojamento de conteúdos gerem as estas infraestruturas. Se for gerida como uma *Cloud*, então as empresas de alojamento de conteúdos podem ser consideradas um provedor de serviços em *Cloud*, se não, são simplesmente uma empresa de alojamento de conteúdos *Web*. (Armbrust, et al., 2009)

### 2.2.3. *Software as a Service (SaaS)*

Este é o nível de maior abstração na *Cloud*. As soluções oferecidas a este nível são soluções aplicacionais já instaladas e prontas a ser utilizadas pelo consumidor final. Estas são geralmente aplicações *on-demand* e *multinancy*<sup>3</sup>. A propriedade *multinancy* indica que existe uma única instância de *software*, apoiada por a infraestrutura do provedor da *Cloud*, que serve de forma simultânea múltiplos clientes organizacionais ou *tenants*. (Armbrust, et al., 2009)

Um exemplo deste tipo de solução, é o oferecido pelo Google através do Gmail no Google App Engine. Nestas soluções o utilizador da *Cloud* não é um programador, mas sim um simples utilizador das soluções aplicacionais oferecidas pelos programadores da *Cloud*. (Armbrust, et al., 2009)

---

<sup>3</sup> Arquitetura na qual uma instância de *software* aplicacional serve múltiplas empresas. Cada empresa é denominada como *tenant*. Os *tenants* podem personalizar determinados conteúdos do *software*, como a cor para o *interface* de utilizador, mas nunca o código fonte aplicacional.

## **2.3. Virtualização**

A virtualização é hoje um tema bastante discutido pelos estudiosos nesta área, e utilizado nas empresas. Contudo, a virtualização já era utilizada na década de 60 pela IBM nos seus sistemas *System/360*. A virtualização é assim um conceito relacionado com a noção de abstração, porque está relacionada com uma camada de abstração sobre um determinado grupo de recursos físicos. Este nível de abstração permite que múltiplos utilizadores ou tarefas tirem partido destes recursos sem a percepção entre eles. (Correia & Sousa, 2010)

O *hypervisor* é um componente que permite que os recursos de *hardware* do sistema sejam utilizados através de uma emulação do *interface* do *hardware*, e que pode ser utilizado por uma máquina virtual num sistema operativo em execução. Deste modo o *hypervisor* partilha os recursos de *hardware* disponíveis com múltiplas máquinas virtuais. (Correia & Sousa, 2010)

### **2.3.1. Tipos de *Hypervisor***

Existem dois tipos de *hypervisor* 's num ambiente de virtualização, o *hypervisor* do tipo 1, também denominado como nativo ou *bare metal*, e o *hypervisor* do tipo 2 também conhecido como *hosted*. (Correia & Sousa, 2010)

No *hypervisor* do tipo 1 existe um maior controlo, flexibilidade e desempenho no ambiente de virtualização, sendo que as máquinas virtuais não estão limitadas às limitações de um sistema operativo. O acesso e comunicação com o *hardware* pela máquina virtual é direto e disponibilizado pelo próprio software de virtualização, proporcionando um maior controlo. Este é o tipo de virtualização, é utilizado na computação em *Cloud* porque permite a partilha de recursos entre múltiplas máquinas virtuais de um único servidor de alta disponibilidade. Os *software* de virtualização do tipo 1 mais utilizados são o Hyper-V da Microsoft e a VMware vSphere da VMware. (Correia & Sousa, 2010)

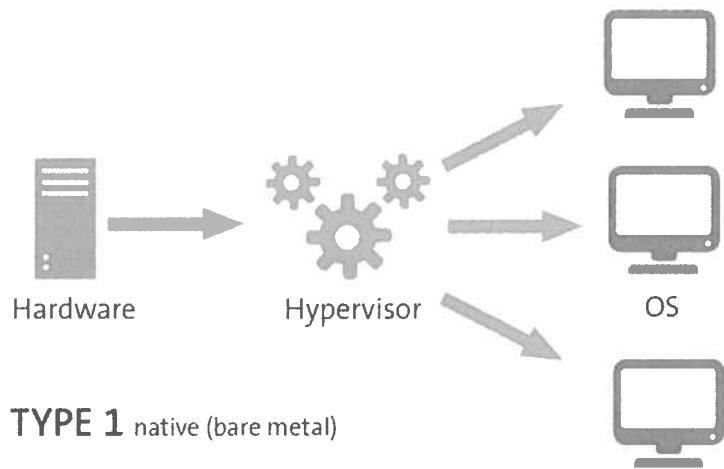


Figura 1: *Hypervisor do Tipo 1.* (Vietstack, 2014)

No *hypervisor* do tipo 2 a máquina virtual é virtualizada sobre o sistema operativo da máquina *host* (hospedeira) que por sua vez está a funcionar diretamente sobre o *hardware*. Nesta arquitetura a máquina virtual comunica de forma indireta com o *hardware* por intermédio do sistema operativo *host*. A máquina virtual é executada através de um *software* de virtualização que permite a criação de múltiplas máquinas virtuais com sistemas operativos diferentes do sistema operativo *host*. Os *software* de virtualização do tipo 2 mais utilizados são a *VirtualBox* da *Oracle* e a *VMware Workstation* da *VMware*. (Correia & Sousa, 2010)

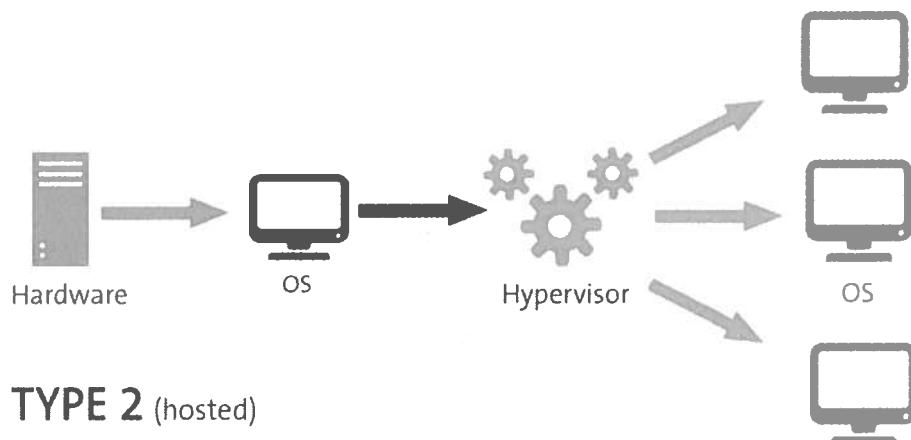


Figura 2: *Hypervisor do Tipo 2.* (Vietstack, 2014)

## 2.4. ADO.NET

A tecnologia ADO.NET pertence à família .NET e permite aos programadores interagir com estruturas de dados de forma desconectada com a fonte de dados. Esta tecnologia permite gerir os dados internos – dados criados em memória que são utilizados somente no interior da aplicação – e os dados externos – que estão alojados fora da aplicação numa base de dados relacional ou num ficheiro de texto. Independentemente da fonte dos dados, o ADO.NET permite congregar esses dados e apresentá-los em tabelas. Além de ADO.NET permitir a manipulação dos dados em formato de tabela, este também permite aceder a fontes de dados que não estejam em formato de tabela. (Patrick, 2010)

Qualquer fonte de dados necessita que seja implementado um conjunto de interfaces específicos do ADO.NET, sendo a finalidade destes, o acesso às fontes de dados. Para tal, os *data provider* são mediadores que ligam o ambiente de desenvolvimento a uma fonte de dados. Cada fonte de dados pode ser suportada por múltiplos *data providers*. (Malik, 2005)

As aplicações desenvolvidas na .NET *framework*<sup>4</sup> dependem das bibliotecas das classes .NET, que existem na forma de ficheiros DLL especiais, e que guardam as funcionalidades de programação permitindo um acesso fácil e rápido aos dados. A maioria das bibliotecas fornecidas com o .NET *framework*, são integradas no código fonte da aplicação quando utilizadas, e são denominadas como *namespaces*. (Patrick, 2010)

### 2.4.1. Componentes principais do ADO.NET

O *namespace System.Data* inclui distintas classes do ADO.NET que trabalham entre elas para fornecer dados em formato de tabela. Esta biblioteca inclui dois grupos principais de classes: o *DataSet* que gere os dados no *software*, e o *Data Provider* que comunica com os sistemas externos das fontes de dados. (Malik, 2005)

---

<sup>4</sup> Ambiente de desenvolvimento de aplicações.

#### **2.4.1.1. *DataSet***

O *DataSet* é constituído por *DataTable*'s e *DataRelation*'s, e funciona em ambiente desconectado da fonte de dados. O *DataTable* funciona sobretudo como uma tabela, e gere todos os dados que estão na aplicação. Cada *DataTable* é constituído por zero ou mais linhas de dados, onde cada linha de dados possui uma coluna com a sua definição. As tabelas definem os itens da  *DataColumn*, que representam individualmente valores de dados. Cada uma das diferentes colunas que constituem as tabelas possui um tipo de dado definido para o tipo de informação que pode conter. Como exemplo, a coluna “Apelido” será do tipo *System.String*, ao passo que a coluna “IVA” será do tipo *System.Decimal*. Outro elemento do *DataTable*, para além dos  *DataColumn*, são as *DataRow*. As *DataRow* são o registo de dados por cada tabela, por cada linha de dados da tabela existe um *DataRow*. O ADO.NET possui métodos que permitem adicionar, apagar, modificar e procurar em cada *DataRow*, da *DataTable*. Existe ainda as *DataRelation* que replicam as relações entre as tabelas existentes na fonte de dados no *DataSet*. (Patrick, 2010)

#### **2.4.1.2. *Data Provider***

A ligação externa às bases de dados é realizada através do *Data Provider*. O ADO.NET possui diferentes tipos de *Data Providers* para comunicar com os diferentes sistemas de gestão de bases de dados e fontes de dados. Existe um *Data Provider* específico para comunicação com o *Microsoft SQL Server*, e para um uso mais genérico existe o ODBC (*Open Database Connecivity*) e o OLE DB (*Object Linking and Embedding Data Base providers*). (Patrick, 2010)

O *Data Povider* é constituído pelos objetos *Connection*, *Command*, *DataAdapter* e *DataReader*. (Malik, 2005) A comunicação com fontes de dados externas ocorre por intermédio do objecto *Connection*. O objeto *Command* representa uma expressão em linguagem SQL para pesquisar, inserir, atualizar e eliminar. Por seu lado, o *DataAdapter* possui ainda definições padrão de consulta para aceder a uma fonte de dados, este constitui a ligação entre a fonte de dados e o *DataSet*. O *DataAdapter* permite ainda efetuar consultas, inserir, atualizar e eliminar informação da fonte de dados original. Todas as alterações no

*DataSet* são depois transmitidas para a base de dados através do *DataAdapter*. Por último o *DataReader* executa somente uma instrução SQL de leitura na base de dados. (Malik, 2005)

### **3. VIRTUAL LIBRARY**

A aplicação desenvolvida em contexto de estudo neste projeto, tem como objetivo aplicar todos os conhecimentos adquiridos ao longo dos 3 anos de duração da Licenciatura Informática. Como tal, foi proposto ao Júri o desenvolvimento de uma biblioteca virtual que transportasse o conceito de biblioteca tradicional para uma biblioteca virtual permitindo a pesquisa, consulta de livros, artigos científicos e teses. A esta aplicação foi dado o nome de *Virtual Library* (Biblioteca Virtual).

#### **3.1. Pré-Requisitos**

As condições impostas para o desenvolvimento da aplicação, são a inclusão de uma base de dados em SQL, um sistema de autenticação próprio, a pesquisa de livros, uma lista de resultados de pesquisa com possibilidade de visualizar um breve detalhe da consulta, a consulta dos livros em formato PDF e um sistema de gestão e administração da aplicação. O acesso a esta aplicação, por intermédio dos utilizadores, é realizado através de uma máquina virtual recorrendo para isso às tecnologias da virtualização.

Assim, os passos para um utilizador visualizar um livro na aplicação ficaram definidos da seguinte forma:<sup>5</sup>

- Acesso à aplicação através do ícone disponível no ambiente de trabalho;
- Autenticação do utilizador;
- Inserir o(s) termo(s) de pesquisa em caixa de texto própria;
- Visualização dos resultados em formato de lista;
- Duplo clique sobre cada linha de resultados oferece um breve detalhe acerca do livro;
- Clicar no botão PDF para consulta do livro.

---

<sup>5</sup> Os passos descritos pressupõem o sucesso da execução de cada passo.

Está ainda disponível no ecrã de pesquisa, após a autenticação do utilizador, uma barra de menu de topo visível mediante as credenciais do utilizador, para efetuar a gestão e administração da aplicação.

Definidas as ideias chaves para o projeto, estabelecidos os pré-requisitos e os objetivos, avançou-se para o desenvolvimento da aplicação.

### 3.2. Estrutura da aplicação

#### 3.2.1. Modelo de Dados

Através dos requisitos apresentados no capítulo “3.1. Pré-Requisitos” na página 25, foi definido a estrutura das tabelas e por consequente o que seria necessário armazenar na base de dados para suportar a aplicação. Assim sendo as tabelas ficaram estruturadas da seguinte forma:

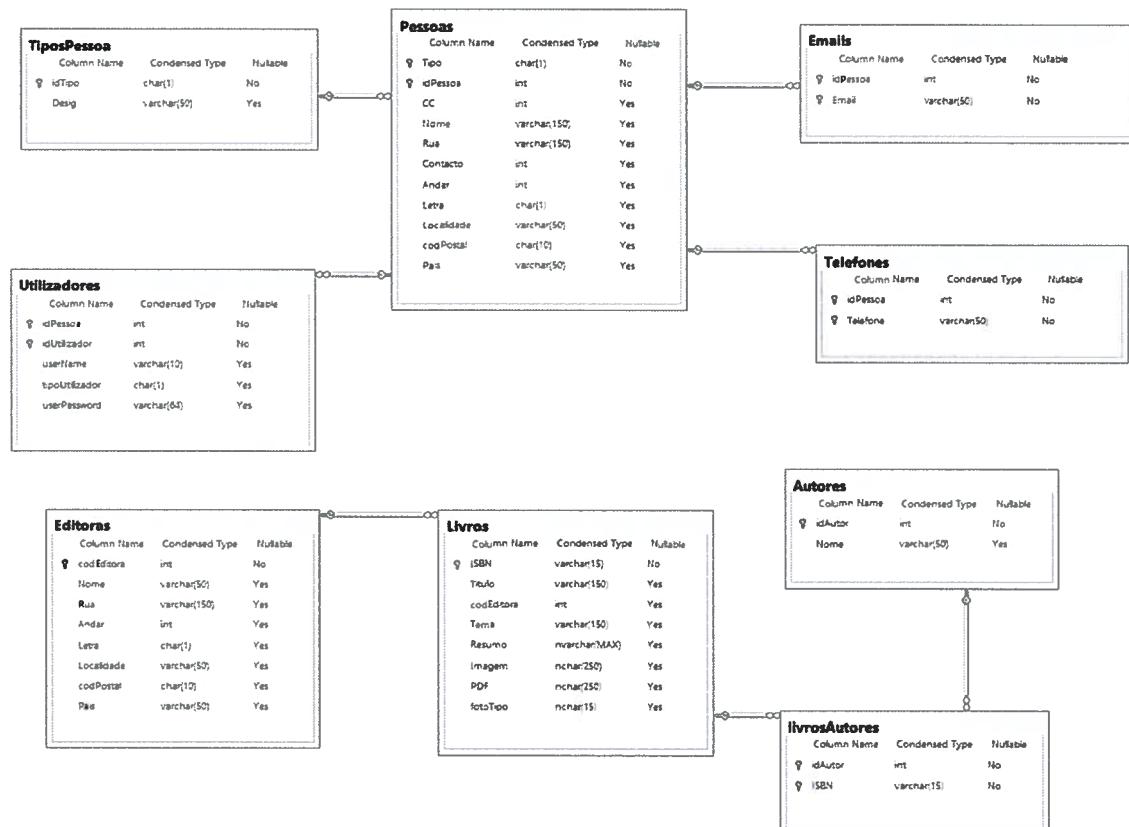


Figura 3: Diagrama da base de dados

Realizando uma análise superficial, podemos verificar que a tabela Pessoas possui como dependentes as tabelas Emails, Telefones e Utilizadores, contudo, a tabela Pessoas também depende por sua vez da tabela TiposPessoa. Estas 5 tabelas gerem entre si a informação dos colaboradores da biblioteca e os utilizadores da aplicação. Por seu lado, a tabela Livros possui como dependentes a tabela livrosAutores dependendo esta diretamente da tabela Autores. A tabela Livros depende ainda da tabela Editoras. Assim podemos concluir que uma editora pode publicar muitos livros, mas um livro apenas pode ter uma editora. Quanto aos autores, um livro pode ter vários autores e vice-versa, sendo esta relação registada na tabela livrosAutores. Vejamos agora a estrutura das tabelas de forma mais detalhada.

### 3.2.1.1. Tabela TiposPessoa

A tabela TiposPessoa é utilizada para registar os tipos de pessoa que podem existir numa organização que gere uma biblioteca. Estes registo são necessários para a criação de registo nas seguintes tabelas.

	idTipo	Desig
1	A	Informática
2	B	Funcionário
3	C	Utilizador

Figura 4: Tabela TiposPessoa

### 3.2.1.2. Tabela Pessoas

Esta tabela guarda a informação geral de cada pessoa, podemos verificar que os registo no campo Tipo assemelham-se aos registo da tabela TiposPessoa, isto acontece porque este campo está dependente dessa tabela. Cada pessoa possui um identificador próprio (idPessoa) que o relaciona com as seguintes tabelas.

	Tipo	idPessoa	CC	Nome	Rua	Contacto	Andar	Letra	Localidade	codPostal	País
1	A	1	[REDACTED]	Carlos Jorge Tregeira Sota	Rua de [REDACTED]	NULL	1	NULL	Montemor-o-Novo	[REDACTED]	Portugal
2	A	2	123456	Joaquim Alfonso Vasconcelos...	Olaia	NULL	NULL	NULL	Lisboa	1234	Portugal
3	C	3	123456	António Manuel Joaquim	Rua da Maria	NULL	1	A	Lisboa	5040-453	Portugal

Figura 5: Tabela Pessoas

### **3.2.1.3. Tabela Emails**

Nesta tabela estão registados os endereços de email, ou correio eletrónico, dos registo na tabela Pessoas. Cada pessoa pode possuir mais que um endereços de email, como tal, esta tabela associa em cada registo o valor do campo idPessoa da tabela Pessoas, que funciona como chave estrangeira.

	<b>idPessoa</b>	<b>Email</b>
1	1	carlo[REDACTED]@hotmail.com
2	2	jafod[REDACTED]@hotmail.com
3	3	amjoaquim@gmail.com

Figura 6: Tabela Emails

### **3.2.1.4. Tabela Telefones**

A tabela Telefones é em tudo idêntica à tabela Emails, esta permite que uma pessoa possua mais que uma forma de contacto, como tal esta associa a cada número o valor do campo idPessoa da tabela Pessoas funcionando como chave estrangeira.

	<b>idPessoa</b>	<b>Telefone</b>
1	1	96[REDACTED]
2	2	91[REDACTED]
3	3	965423658

Figura 7: Tabela Telefones

### 3.2.1.5. Tabela Utilizadores

A tabela Utilizadores possui os registo de autenticação na aplicação *Virtual Library* de cada utilizador. Para efetuar um registo de utilizador é necessário associar este registo ao campo *idPessoa* da tabela Pessoas e definir o tipo de utilizador através dos registo na tabela TiposPessoa. Os campos *userName* e *userPassword* são as credenciais de *login* definidas para cada utilizador. De referir ainda que o campo *userPassword* possui a palavra-passe do utilizador encriptada com SHA-256<sup>6</sup>.

	<i>idPessoa</i>	<i>idUtilizador</i>	<i>userName</i>	<i>tipoUtilizador</i>	<i>userPassword</i>
1	1	1	csota	A	[REDACTED] 3ca7eb0c6a6c6b...
2	2	2	jafonso	A	f91a799a[REDACTED] ...
3	3	3	amjoaquim	C	46070d4b[REDACTED] 322444900a435...

Figura 8: Tabela Utilizadores

### 3.2.1.6. Tabela Editoras

A tabela Editoras guarda a informação das editoras dos livros presentes na aplicação. Os campos que compõem esta tabela são campos de informação geral. O campo *codEditora* é a chave primária da tabela e é incrementado automaticamente sempre que uma nova editora é adicionada.

	<i>codEditora</i>	<i>Nome</i>	<i>Rua</i>	<i>Andar</i>	<i>Letra</i>	<i>Localidade</i>	<i>codPostal</i>	<i>Pais</i>
1	1	Sams Publishing	800 East 96th Street	NULL	NULL	Indianapolis	46240	USA
2	2	Sybex	111 River Street	NULL	NULL	Hoboken	7030	USA
3	3	Apress	233 Spring Street	NULL	NULL	New York	10013	USA
4	4	Wiley Publishing, Inc.	10475 Crosspoint Boulevard	NULL	NULL	Indianapolis	46256	USA
5	5	O'Reilly Media, Inc.	1005 Gravenstein Highway North	NULL	NULL	Sebastopol	95472	USA
6	6	John Wiley & Sons, Inc.	10475 Crosspoint Boulevard	NULL	NULL	Indianapolis	46256	USA

Figura 9: Tabela Editoras

<sup>6</sup> Secure Hash Algorithm;

### 3.2.1.7. Tabela Autores

Esta é uma tabela possui apenas dois campos, o idAutor e o Nome. O campo idAutor é a chave primária da tabela e é incrementado automaticamente sempre que um novo registo é realizado. O campo Nome guarda o nome do autor.

	<b>idAutor</b>	<b>Nome</b>
1	1	Rand Morimoto
2	2	Michael Noel
3	3	Omar Droubi
4	4	Ross Mistry
5	5	Chris Amaris
6	6	Jeremy Moskowitz
7	7	Andrew Troelsen

Figura 10: Tabela Autores

### 3.2.1.8. Tabela livrosAutores

A tabela livrosAutores guarda os autores de cada livro, ou se preferimos, relaciona o autor com os livros. Sabendo que um livro pode ter um ou vários autores, esta tabela utiliza como chave primária da tabela Autores e da tabela Livros, os campos idAutor e ISBN respetivamente, para a associação entre os autores e livros.

	<b>idAutor</b>	<b>ISBN</b>
1	1	978-0672330926
2	2	978-0672330926
3	3	978-0672330926
4	4	978-0672330926
5	5	978-0672330926
6	6	978-1118289402
7	7	9781590598849
8	8	978-0470506387
9	9	978-0735638884
10	10	978-1118336922
11	11	978-0596520830
12	12	978-1118176719

Figura 11: Tabela livrosAutores

### 3.2.1.9. Tabela Livros

Por último, a tabela Livros possui a informação dos livros. A chave primária desta tabela é o campo ISBN estando este relacionado com a tabela livrosAutores. O campo codEditora é a chave estrangeira desta tabela e está relacionado com a tabela Editoras. Por último os campos Imagem e PDF guardam o nome da capa e do ficheiro PDF do livro.

	ISBN	Titulo	codEditora	Tema	Resumo	Imagen	PDF	fotoTipo
1	978-0470506387	Cloud Computing with the Wi...	4	NULL	With the Azure Services ...	978-0470506387.jpg	978-0470506387.pdf	NULL
2	978-0596520830	Learning SQL	5	NULL	Programming languages ...	978-0596520830.jpg	978-0596520830.pdf	NULL
3	978-0672330926	Windows Server 2008 R2 Unl...	1	NULL	Windows Server 2008 R...	978-0672330926.jpg	978-0672330926.pdf	NULL
4	978-0735638884	Microsoft ADO.NET 4 Step b...	5	NULL	Teach yourself the funda...	978-0735638884.jpg	978-0735638884.pdf	NULL
5	978-1118176719	Virtualization Essentials	2	NULL	A full-color beginner's gui...	978-1118176719.jpg	978-1118176719.pdf	NULL
6	978-1118289402	Group Policy: Fundamentals, ...	2	NULL	Freshly updated to includ...	978-1118289402.jpg	978-1118289402.pdf	NULL
7	978-1118289426	Mastering Windows Server 2...	2	NULL	Check out the new Hype...	978-1118289426.jpg	978-1118289426.pdf	NULL
8	978-1118336922	Beginning Object-Oriented Pr...	6	NULL	Wrox beginners' guides h...	978-1118336922.jpg	978-1118336922.pdf	NULL
9	978-1449320102	C# 5.0 in a Nutshell: The Defi...	5	NULL	When you have a questi...	978-1449320102.jpg	978-1449320102.pdf	NULL
10	97815905988-49	Pro C# 2008 and the .NET 3....	3	NULL	In this last version, .NET ...	97815905988-49.jpg	97815905988-49.pdf	NULL

Figura 12: Tabela Livros

### 3.3. Desenvolvimento

A aplicação foi desenvolvida recorrendo ao IDE (*Integrated Development Environment*) Visual Studio Ultimate 2012, sendo utilizado a linguagem de programação C# e a tecnologia ADO.NET. O projeto desta aplicação foi criado em Windows *Presentation Foundation client application* (WPF). A base de dados da aplicação foi criada e adicionada ao projeto em Visual Studio de forma a facilitar o desenvolvimento da aplicação. Numa fase final, e quando estiver concluído o desenvolvimento, os ficheiros com extensão .mdf<sup>7</sup> e .ldf<sup>8</sup> serão adicionados ao SQL Server 2012.

O *solution explorer* é uma janela do Visual Studio que funciona de forma similar ao conhecido Explorador do Windows, aqui podemos ver todos os itens que constituem o projeto de desenvolvimento.

<sup>7</sup> Extensão do ficheiro de base de dados SQL;

<sup>8</sup> Extensão do ficheiro de log da base de dados SQL;



Figura 13: *Solution Explorer* da aplicação

Analisando a Figura 13 em epígrafe, verificamos que o projeto é constituído por 11 janelas, sendo que estas não se sobrepõem, por forma a facilitar a navegação do utilizador, e por uma classe. É ainda possível verificar que existem 5 pastas, sendo estas:

- *bookImages* - guarda a capa dos livros que estejam na aplicação;
- *Books* - guarda os livros em formato PDF;
- *Database* – possui a base de dados e o *dataSet* da aplicação;
- *Icons* – pasta utilizada para guardar os ícones da aplicação;
- *Images* – pasta utilizada para guardar as imagens da aplicação.

Os próximos subcapítulos analisam de forma mais concreta as janelas e a classe da aplicação.

### 3.3.1. Class Cripto

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Security.Cryptography;
using System.Text;
using System.Threading.Tasks;

namespace Biblioteca
{
    public static class Cripto
    {
        public static string sha256(string password)
        {
            SHA256Managed crypt = new SHA256Managed();
            string hash = String.Empty;
            byte[] crypto = crypt.ComputeHash(Encoding.ASCII.GetBytes(password), 0,
Encoding.ASCII.GetByteCount(password));
            foreach (byte bit in crypto)
            {
                hash += bit.ToString("x2");
            }
            return hash;
        }
    }
}
```

Figura 14: Classe Cripto.

A classe Cripto tem como objetivo receber um argumento do tipo *string* e retornar o argumento recebido encriptado com SHA256. Para tal, é necessário importar o *namespace* *System.Security.Cryptography* de forma a poder ser possível criar um objeto do tipo *SHA256Managed* e instanciá-lo. A classe Cripto e o método sha256 são do tipo *public* sendo desta forma possível aceder a este método através da classe Cripto em qualquer janela da aplicação.

Quando o método sha256 é chamado, este recebe um argumento do tipo *string*, cria um objeto do tipo *SHA256Managed* e duas variáveis, uma *string* e um *array* de *byte*. Este *array* vai guardar em cada posição uma *hash* através do método *crypt.ComputeHash()* que

percorre e encripta todos os bytes. Por último o ciclo *foreach* percorre e devolve todos os bytes do array cripto convertidos em string.

### 3.3.2. *MainWindow*



Figura 15: Janela *MainWindow*

Esta é a janela inicial da aplicação, e pressupõe que o utilizador preencha as caixas de texto com o seu utilizador e palavra-passe de forma a autenticar-se e aceder à aplicação. Debruçando um olhar sobre o código, é necessário importar alguns *namespaces*, além dos habituais, para tornar possível a comunicação com a base de dados e com o *DataSet*.

```
using System.Data;
using System.Data.SqlClient;
using System.IO;
using Biblioteca.DataBase.dataSetBibliotecaTableAdapters;
using Biblioteca.DataBase;
```

Figura 16: *Namespaces* da janela *MainWindow*

Os dois botões presentes nesta janela quando clicados acionam um *event handler*. O botão alterar *password* aciona o *event handler* *btnPassLost\_Click\_1* que abre uma nova

janela que permite ao utilizador alterar a sua palavra-passe, vejamos este de forma mais detalhada.

```
private void btnPassLost_Click_1(object sender, RoutedEventArgs e)
{
    if (txtUser.Text.Length > 0)
    {
        dataTableUtilizadores =
tableAdapterUtilizadores.getByName(txtUser.Text.ToString());
        if (dataTableUtilizadores.Rows.Count > 0)
        {
            string userName = txtUser.Text;
            PasswordRecover passRecover = new PasswordRecover(userName);
            txtPassword.Clear();
            passRecover.ShowDialog();
        }
        else
        {
            lblMsg.Visibility = Visibility.Visible;
            lblMsg.Content = "Introduza um utiliador válido.";
        }
    }
    else
    {
        lblMsg.Visibility = Visibility.Visible;
        lblMsg.Content = "Preencha o campo Username.";
    }
}
```

Figura 17: Event handler `btnPassLost_Click_1`

O evento `btnPassLost_Click_1` verifica se o campo *Username* está preenchido, caso não esteja, surge uma mensagem ao utilizador a solicitar o preenchimento do campo *Username*. Se o campo estiver preenchido, é efetuado uma consulta à base de dados com o nome de utilizador introduzido, caso o nome não corresponda a nenhum utilizador, o utilizador é notificado para introduzir um utilizador válido. Se este utilizador for válido, é aberto a janela de recuperação da palavra-passe que permanecerá ativa enquanto aberta não permitindo que o utilizador regresse à janela principal.

```

private void btnLogin_Click_1(object sender, RoutedEventArgs e)
{
    if (txtUser.Text.Length > 0)
    {
        dataTableUtilizadores =
tableAdapterUtilizadores.getDataByUserName(txtUser.Text.ToString());
        tbUtilizadoresRow =
(dataSetBiblioteca.UtilizadoresRow)dataTableUtilizadores[0];
        string typedPassword = Cripto.sha256(txtPassword.Password);
        string storedPassword = tbUtilizadoresRow.userPassword;
        if (string.Compare(typedPassword, storedPassword) == 0)
        {
            dataTableUtilizadores.Clear();
            string userName = txtUser.Text;
            Home homepage = new Home(userName);
            homepage.Show();
            this.Close();
        }
        else
        {
            lblMsg.Visibility = Visibility.Visible;
            lblMsg.Content = "Utilizador ou palavra-passe incorrecta.";
        }
    }
    else
    {
        lblMsg.Visibility = Visibility.Visible;
        lblMsg.Content = "Preencha os campos Username e Password.";
    }
}

```

Figura 18: Event handler `btnLogin_Click_1`

Em relação ao botão “Iniciar sessão” este receber um *click* por parte do utilizador, aciona o *event handler* `btnLogin_Click_1`. Este evento verifica inicialmente se o campo Username está preenchido, em caso negativo surge o utilizador é notificado. Se o campo estiver preenchido, é efetuado uma consulta à tabela Utilizadores para verificar se existe algum utilizador que corresponda ao introduzido pelo utilizador, de seguida é atribuído à variável *typedPassword* a encriptação da palavra-passe introduzida pelo utilizador através do método `sha2569` da classe Cripto, é ainda atribuído à variável *storedPassword* a palavra-passe do utilizador, que está encriptada, guardada na base de dados na tabela Utilizadores. Seguidamente é comparada a palavra-passe introduzida com a palavra-passe que está em base de dados e se estas não coincidirem, o utilizador é notificado, caso sejam idênticas, a janela *MainWindow* é fechada e é aberta a janela *Home* da aplicação.

---

<sup>9</sup> Método revisto na página 33.

### 3.3.3. passwordRecover

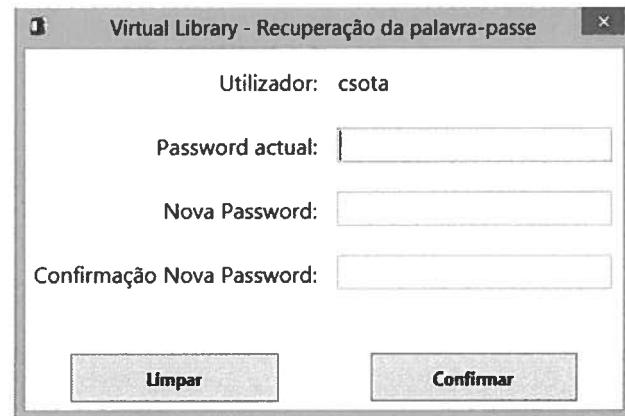


Figura 19: Janela passwordRecover

Nesta janela o utilizador tem a possibilidade de alterar a sua palavra-passe mediante a introdução da palavra-passe atual. Como medida de segurança, é necessário que o utilizador confirme a nova palavra-passe introduzida de forma a evitar lapsos. Tal como sucedeu na janela inicial, é necessário importar os mesmos *namespaces*<sup>10</sup>.

```
public PasswordRecover(string _userName)
{
    InitializeComponent();
    txtPassword.Focus();
    lblUser.Content = userName = _userName;
}
```

Figura 20: Método *PasswordRecover*

Voltando um pouco atrás à figura 17 da página 35, foi criada uma instância desta página e foi passado um argumento do tipo *string*. Este argumento é o nome de utilizador que é atribuído à *Label* com o nome *lbluser*.

---

<sup>10</sup> Ver figura 16 na página 34.

```

private void btnClean_Click_1(object sender, RoutedEventArgs e)
{
    txtPassword.Clear();
    txtNewPassword.Clear();
    txtNewPassword1.Clear();
}

```

Figura 21: *Event Handler* btnClean\_Click\_1

O botão Limpar possui o *event handler* btnClean\_Click\_1 associado. Este evento limpa os dados introduzidos nas três caixas de texto passíveis de preenchimento.

```

private void btnConfirm_Click_1(object sender, RoutedEventArgs e)
{
    if (txtPassword.Password.Length > 0 && txtNewPassword.Password.Length > 0 &&
    txtNewPassword1.Password.Length > 0)
    {
        if (txtNewPassword.Password == txtNewPassword1.Password)
        {
            dataTableUtilizadores =
tableAdapterUtilizadores.getDataByUserName(lblUser.Content.ToString());
            tbUtilizadoresRow =
(dataSetBiblioteca.UtilizadoresRow)dataTableUtilizadores.Rows[0];
            string typedPassword = Cripto.sha256(txtPassword.Password);
            string storedPassword = tbUtilizadoresRow.userPassword;
            if (string.Compare(typedPassword, storedPassword) == 0)
            {
                string password = Cripto.sha256(txtNewPassword.Password);
                tbUtilizadoresRow.userPassword = password;
                tableAdapterUtilizadores.Update(dataTableUtilizadores);
                dataTableUtilizadores.AcceptChanges();
                MessageBoxResult msgSuccess = MessageBox.Show("Palavra-passe
alterada com sucesso!", "Confirmação");
                this.Close();
            }
            else
            {
                lblMsg.Visibility = Visibility.Visible;
                lblMsg.Content = "Palavra-passe incorrecta.";
            }
        }
        else
        {
            lblMsg.Visibility = Visibility.Visible;
            lblMsg.Content = "Os dados introduzidos estão incorrectos.";
        }
    }
    else
    {
        lblMsg.Visibility = Visibility.Visible;
        lblMsg.Content = "É necessário preencher todos os campos.";
    }
}

```

Figura 22: *Event Handler* btnConfirm\_Click\_1

O botão Confirmar possui o *event handler* btnConfirm\_Click\_1 associado, sendo que, numa fase inicial verifica se os três campos de preenchimento estão preenchidos, em caso negativo, o utilizador é notificado para preencher todos os campos. Numa segunda fase, será comparado a nova palavra-passe com a confirmação da mesma, no caso de serem diferentes, mais uma vez o utilizador é alertado. Em caso de sucesso, é efetuada uma consulta à base de dados na tabela Utilizadores pelo nome de utilizador em questão. De seguida será usado o método sha256<sup>11</sup> da classe Cripto para encriptar a palavra passe atual do utilizador e comparar com a que está encriptada na base de dados, em caso de falha surge mais uma vez uma notificação para o utilizador. Se estas coincidirem, é utilizado o método sha256 novamente para encriptar a nova palavra-passe e guardar as alterações na tabela Utilizadores na linha referente ao utilizador em questão. Por último, o utilizador é notificado com a confirmação do sucesso desta operação, sendo a janela *passwordRecover* fechada voltando o utilizador ao ecrã inicial.

### 3.3.4. Home

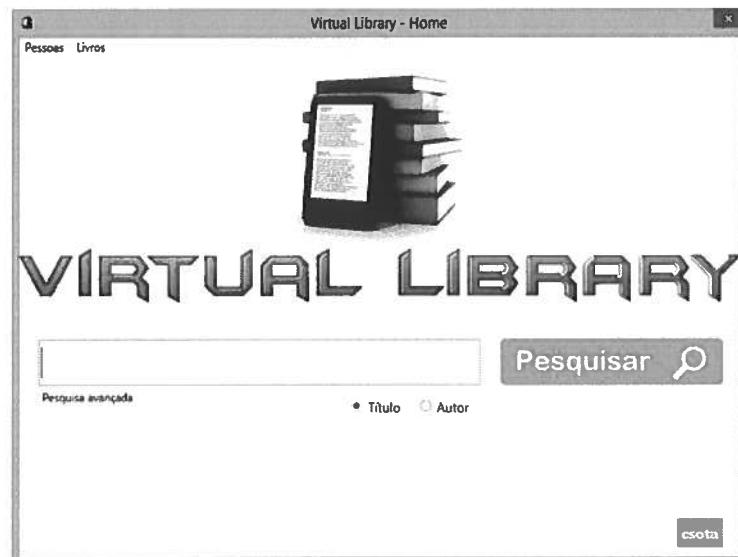


Figura 23: Janela *Home*

---

<sup>11</sup> Método revisto na página 33.

Nesta janela o utilizador pode efetuar pesquisas por título ou autor, existindo ainda a possibilidade de efetuar uma pesquisa avançada. A barra de menu no topo da janela possibilita a administração e gestão da aplicação, esta barra apenas está visível e disponível para utilizadores com acesso para tal. Mais uma vez, em todas as janelas é necessário importar os *namespaces*, sendo esta uma rotina o leitor deverá pressupor que em todas a janelas estes estão presentes, pelo que não serão referidos no futuro.

```
private void menuItemTiposPessoa_Click_1(object sender, RoutedEventArgs e)
{
    tipoPessoa tipoPessoa = new tipoPessoa(userName);
    tipoPessoa.Show();
    this.Close();
}

private void menuItemUtilizadores_Click_1(object sender, RoutedEventArgs e)
{
    Utilizadores users = new Utilizadores(userName);
    users.Show();
    this.Close();
}

private void menuItemPessoas_Click_1(object sender, RoutedEventArgs e)
{
    Pessoas pessoas = new Pessoas(userName);
    pessoas.Show();
    this.Close();
}

private void menuItemEditoras_Click_1(object sender, RoutedEventArgs e)
{
    Editoras editoras = new Editoras(userName);
    editoras.Show();
    this.Close();
}

private void menuItemLivros_Click_1(object sender, RoutedEventArgs e)
{
    Livros livros = new Livros(userName);
    livros.Show();
    this.Close();
}

private void menuItemAutores_Click_1(object sender, RoutedEventArgs e)
{
    Autores autores = new Autores(userName);
    autores.Show();
    this.Close();
}
```

Figura 24: Event handler's da barra de navegação de topo

Na figura 24 podemos observar os *event handler's* que estão associados às diferentes opções disponíveis na barra de menu no topo. Um clique numa destas opções disponíveis, e a janela *Home* é fechada sendo aberto a nova janela correspondente juntamente com um argumento com a identificação do utilizador. As opções disponíveis são:

#### Pessoas

- Pessoas;
- Categorias;
- Utilizadores.

#### Livros

- Livros;
- Editoras;
- Autores.

```
UtilizadoresTableAdapter tableAdapterUtilizadores = new  
UtilizadoresTableAdapter();  
dataSetBiblioteca.UtilizadoresDataTable dataTableUtilizadores;  
dataSetBiblioteca.UtilizadoresRow tbUtilizadoresRow;  
  
LivrosTableAdapter tableAdapterLivros = new LivrosTableAdapter();  
dataSetBiblioteca.LivrosDataTable dataTableLivros;  
dataSetBiblioteca.LivrosRow tbLivrosRow;  
  
private SqlConnection sqlConn = new SqlConnection();  
private System.Data.DataSet dataSet = new System.Data.DataSet();  
private System.Data.DataTable dataTable;  
  
string userName;  
int searchOption;
```

Figura 25: Variáveis globais da janela *Home*

```
public Home(string _userName)  
{  
    InitializeComponent();  
    userName = _userName;  
    lblUser.Content = userName;  
    txtSearch.Focus();  
    dataTableUtilizadores = tableAdapterUtilizadores.getDataByUserName(userName);  
    tbUtilizadoresRow =  
(dataSetBiblioteca.UtilizadoresRow)dataTableUtilizadores.Rows[0];  
    if (tbUtilizadoresRow.tipoUtilizador.Equals("C"))  
    {  
        dockPanel.Visibility = Visibility.Hidden;  
    }  
}
```

Figura 26: Método *Home*

O método *Home* coloca o *focus* do cursor do teclado na caixa de texto de pesquisa quando a janela *Home* é iniciada, preenche o *dataTableUtilizadores* com uma consulta à base de dados na tabela *Utilizadores* para verificar o tipo de utilizador que possui a sessão iniciada. Caso este tipo corresponda à letra ‘C’, este utilizador não tem permissão de administração na aplicação, assim sendo a barra de menu no topo não está visível. Qualquer outro tipo de utilizador, ‘A’ ou ‘B’, possui acesso a esta barra de administração.

```
private SqlCommand search(string searchParam, int searchOption)
{
    SqlCommand command = new SqlCommand();
    string sql;
    string[] allWords = searchParam.Split(new char[] { ' ' },
    StringSplitOptions.RemoveEmptyEntries);
    if (searchOption == 1)
    {
        sql = "SELECT Livros.ISBN, Livros.Titulo, Livros.Tema, Livros.Resumo,
Livros.Imagem, Livros.fotoTipo, Livros.PDF, Editoras.Nome AS nomeEditora FROM
Livros INNER JOIN Editoras ON Livros.codEditora = Editoras.codEditora WHERE ";
    }
    else
    {
        sql = "SELECT Livros.ISBN, Livros.Titulo, Livros.Tema, Livros.Resumo,
Livros.Imagem, Livros.fotoTipo, Livros.PDF, Editoras.Nome AS nomeEditora FROM
Livros INNER JOIN livrosAutores ON Livros.ISBN = livrosAutores.ISBN INNER JOIN
Autores ON livrosAutores.idAutor = Autores.idAutor INNER JOIN Editoras ON
Livros.codEditora = Editoras.codEditora WHERE ";
    }
    using (command)
    {
        for (int i = 0; i < allWords.Length; ++i)
        {
            if (i > 0)
            {
                sql += "OR ";
            }
            if (searchOption == 1)
            {
                sql += string.Format("(Livros.Titulo LIKE '%{0}%') ", 
allWords[i]);
            }
            else
            {
                sql += string.Format("(Autores.Nome LIKE '%{0}%') ", 
allWords[i]);
            }
        }
        command.CommandText = sql;
    }
    return command;
}
```

Figura 27: Método *search*

O método *search* permite transformar o parâmetro de pesquisa introduzido pelo utilizador numa instrução SQL executável e guardar esta numa variável do tipo *SqlCommand*. Desta forma este método recebe dois argumentos, o parâmetro de pesquisa introduzido pelo utilizador e um inteiro com a informação se a pesquisa a realizar é por título ou por autor.

O parâmetro de pesquisa é convertido num *array* de *strings* para permitir uma pesquisa mais eficaz e com melhores resultados, assim por cada espaço detetado neste é criado uma nova posição neste *array* com a palavra após o espaço. Para demonstrar o porquê desta opção, imaginemos que existe um livro com o título “Windows Server 2008 R2 Unleashed”, e o utilizador pesquisa por “Windows Server” ou “Server 2008”, este parâmetro seria introduzido na cláusula *WHERE* da instrução SQL e encontrava com sucesso o livro. Contudo, se o utilizador pesquisa por “Windows 2008” ou “Windows Unleashed” este não retornaria qualquer resultado. Assim sendo, este método permite separar as palavras do parâmetro de pesquisa e guardar cada palavra numa posição do array *allWords*. O próximo passo deste método é verificar se a pesquisa é feita por título ou autor, para isso é verificado o valor do argumento *searchOption* e é escolhido o comando SQL apropriado. Por último é usado um ciclo para percorrer todas as posições do array *allWords* e completar o comando SQL com a informação de pesquisa. No final este método retorna o comando SQL pronto a ser executado.

A figura seguinte demonstra um exemplo do comando SQL retornado pelo método *search* para uma pesquisa por “Windows group policy 2012”.

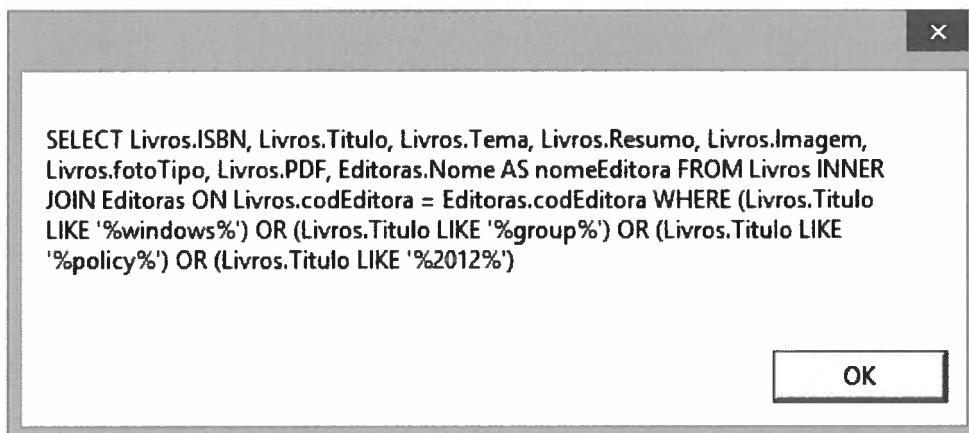


Figura 28: Exemplo do comando SQL retornado pelo método *search*

```

private void btnSearch_MouseLeftButtonUp_1(object sender, MouseButtonEventArgs e)
{
    if (txtSearch.Text.Trim().Length > 0)
    {
        if(txtSearch.Text.Trim().Length > 3)
        {
            if (rbSearchOption1.IsChecked.Value)
            {
                searchOption = 1;
                sqlConn.ConnectionString =
Properties.Settings.Default.BibliotecaConnectionString;
                string connectionString = sqlConn.ConnectionString.ToString();
                SqlDataAdapter sqlDataAdapter = new
SqlDataAdapter(search(txtSearch.Text, searchOption).CommandText,
connectionString);
                sqlDataAdapter.Fill(dataSet, " booksResult ");
                dataTable = dataSet.Tables["booksResult "];
            }
            else
            {
                searchOption = 2;
                sqlConn.ConnectionString =
Properties.Settings.Default.BibliotecaConnectionString;
                string connectionString = sqlConn.ConnectionString.ToString();
                SqlDataAdapter sqlDataAdapter = new
SqlDataAdapter(search(txtSearch.Text, searchOption).CommandText,
connectionString);
                sqlDataAdapter.Fill(dataSet, "booksResult ");
                dataTable = dataSet.Tables["booksResult "];
            }
            if (dataTable.Rows.Count > 0)
            {
                searchResults searchResults = new searchResults(userName,
search(txtSearch.Text, searchOption));
                searchResults.Show();
                this.Close();
            }
            else
            {
                MessageBox.Show("A pesquisa por '"+ txtSearch.Text +"' não
retornou quaisquer resultados","Aviso");
            }
        }
        else
        {
            lblMsg.Content = "Elemento de pesquisa introduzido insuficiente.";
            lblMsg.Visibility = Visibility.Visible;
        }
    }
    else
    {
        lblMsg.Content = "É necessário preencher o campo de pesquisa.";
        lblMsg.Visibility = Visibility.Visible;
    }
}

```

Figura 29: Event handler `btnSearch_MouseLeftButtonUp_1`

Quando o utilizador introduz um parâmetro de pesquisa e clica no botão “Pesquisar”, é acionado o *event handler* *btnSearch\_MouseLeftButtonUp\_1*, este evento inicia dois testes ao parâmetro de pesquisa introduzido. O primeiro teste verifica se a caixa de texto de pesquisa está preenchida, caso não esteja, notifica o utilizador, o segundo teste exige que o parâmetro de pesquisa possua mais de 3 caracteres para garantir um resultado de pesquisa confiável e preciso, se o parâmetro de pesquisa não compreender este requisito, mas uma vez o utilizador é notificado. O próximo passo verifica qual a *checkbox* ativa determinando se a pesquisa é realizada por “Título” ou por “Autor”. Mediante esta condição, é criado o *DataAdapter* através da *connectionString* de ligação à base de dados, e do método *search*<sup>12</sup>, método que recebe como argumentos o termo de pesquisa e o tipo de pesquisa (título ou autor). Neste momento o *DataAdapter* possui o comando SQL para efetuar a consulta à base de dados e a *connectionString* de ligação, sendo agora possível guardar o resultado da pesquisa no *dataSet* numa tabela com o nome “*booksResult*”. Por fim, e antes da janela com os resultados da pesquisa ser apresentada, importa verificar se a pesquisa efetuada retornou quaisquer resultados, para isso é utilizado a propriedade *Rows.Count* do *dataTable* para determinar o número de linhas que a tabela possui. Se número de linhas for superior a 0 a janela *Home* é fechada e é aberta a janela *searchResults* da aplicação, caso contrário o utilizador é notificado que a pesquisa não retornou resultados.

```
private void lblAdvancedSearch_PreviewMouseLeftButtonDown_1(object sender,
    MouseEventArgs e)
{
    advancedSearch advancedSearch = new advancedSearch(userName);
    advancedSearch.Show();
    this.Close();
}
```

Figura 30: *Event handler* *lblAdvancedSearch\_PreviewMouseLeftButtonDown\_1*

Como referido anteriormente é possível efetuar uma pesquisa avançada, para isso o utilizador deve clicar em “Pesquisa avançada” na janela *Home*. Este dispara um *event handler* que abre uma nova janela. Este evento é idêntico a outros revistos anteriormente.

---

<sup>12</sup> Método revisto na página 41.

### 3.3.5. Advanced Search



Figura 31: Janela *Advanced Search*

Esta janela permite efetuar uma pesquisa avançada e personalizada, indo ao encontro das necessidades do utilizador. Permite a pesquisa por título, autor e editora, sendo possível combinações de “E” e “OU” entre os três campos. As similaridades com a janela *Home* são inúmeras, desde já o *event handler* `btnSearch_MouseLeftButtonUp_1` associado ao botão “Pesquisar” é idêntico. Tendo este fator em consideração, vamos concentrarmo-nos no que distingue esta janela da anterior.

```
private string[] searchArray(string searchParam)
{
    string[] allWords = searchParam.Split(new char[] { ' ' },
StringSplitOptions.RemoveEmptyEntries);
    return allWords;
}
```

Figura 32: Método *searchArray*

O método *searchArray* recebe o parâmetro de pesquisa introduzido pelo utilizador e por cada espaço detetado neste é criado uma posição neste *array* apenas com a palavra. Como existem três caixas de texto distintas, este método será utilizado por cada caixa de texto preenchida.

```
private SqlCommand search()
{
    SqlCommand command = new SqlCommand();
    string sql = "SELECT DISTINCT Livros.ISBN, Livros.Titulo, Livros.Tema,
Livros.Resumo, Livros.Imagem, Livros.fotoTipo, Livros.PDF, Editoras.Nome AS
nomeEditora FROM Livros INNER JOIN livrosAutores ON Livros.ISBN =
livrosAutores.ISBN INNER JOIN Autores ON livrosAutores.idAutor = Autores.idAutor
INNER JOIN Editoras ON Livros.codEditora = Editoras.codEditora WHERE (";
    string[] titulo = searchArray(txtTitulo.Text);
    string[] autor = searchArray(txtAutor.Text);
    string[] editora = searchArray(txtEditora.Text);
    using (command)
    {
        if (txtTitulo.Text.Trim().Length > 0)
        {
            for (int i = 0; i < titulo.Length; ++i)
            {
                if (i > 0)
                {
                    sql += "OR ";
                }
                sql += string.Format("(Livros.Titulo LIKE '%{0}%') ", 
    titulo[i]);
            }
        }
        if (txtAutor.Text.Trim().Length > 0)
        {
            if (cbOption1.SelectedIndex == 0 && txtTitulo.Text.Trim().Length >
0)
                sql += ") AND (";
            if (cbOption1.SelectedIndex == 1 && txtTitulo.Text.Trim().Length >
0)
                sql += ") OR (";
            for (int i = 0; i < autor.Length; ++i)
            {
                if (i > 0)
                {
                    sql += "OR ";
                }
                sql += string.Format("(Autores.Nome LIKE '%{0}%') ", 
    autor[i]);
            }
        }
        if (txtEditora.Text.Trim().Length > 0)
        {
            if (cbOption2.SelectedIndex == 0 && (txtAutor.Text.Trim().Length >
0 || txtTitulo.Text.Trim().Length > 0))
                sql += ") AND (";
```

```

if (cbOption2.SelectedIndex == 1 && (txtAutor.Text.Trim().Length > 0 ||
txtTitulo.Text.Trim().Length > 0))
    sql += ") OR (";
    for (int i = 0; i < editora.Length; ++i)
    {
        if (i > 0)
        {
            sql += "OR ";
        }
        sql += string.Format("(Editoras.Nome LIKE '{0}%') ", 
editora[i]);
    }
    sql += string.Format("(Editoras.Nome LIKE '{0}%') ", 
editora[i]);
}
sql += ")";
command.CommandText = sql;
}
return command;
}

```

Figura 33: Método *search*

O método *search* constrói o comando SQL mediante os parâmetros de pesquisa introduzidos nas três caixas de texto pelo utilizador. Para isso este método começa por recorrer ao método *searchArray* para criar um *array* com o(s) parâmetro(s) de pesquisa por cada uma das três caixas de texto que estejam preenchidas. Seguidamente é testado se a caixa de texto referente ao título possui texto introduzido, se possuir, um ciclo irá percorrer todas as posições do *array* *titulo* e introduzir cada elemento de pesquisa no comando SQL. O passo seguinte verifica se a caixa de texto referente ao Autor está preenchida, caso esteja, verifica agora qual a opção de combinação selecionada (“E” ou “OU”) e se o campo título está preenchido, se ambas as condições forem verdadeiras introduz no comando SQL a combinação selecionada. Novamente um ciclo irá percorrer todas as posições do *array* *autor* e introduzir cada elemento de pesquisa no comando SQL. Falta agora verificar se a caixa de texto *editora* está preenchida, verificar qual a opção de combinação selecionada (“E” ou “OU”) e determinar se o campo, título ou autor, estão preenchidos. Se todas as condições forem verdadeiras introduz no comando SQL a combinação selecionada. Mais uma vez um ciclo irá percorrer todas as posições do *array* *editora* e introduzir cada elemento de pesquisa no comando SQL. Por último este método retorna o comando SQL finalizado e pronto a ser executado.

### 3.3.6. Search Results

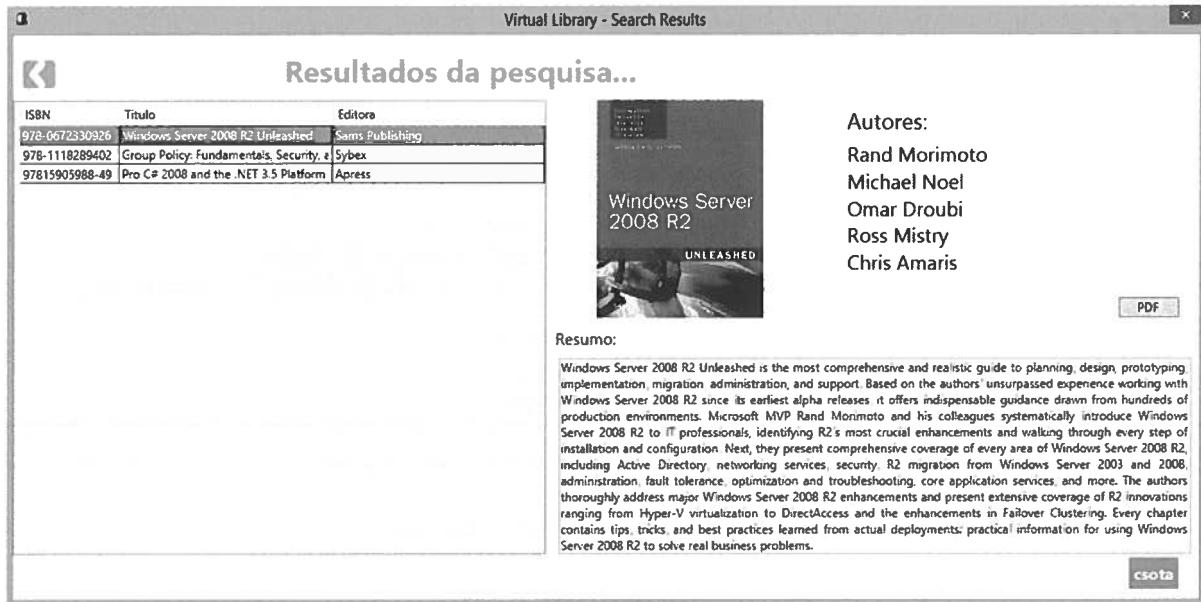


Figura 34: Janela *searchResults*

A janela *searchResults* apresenta os resultados da pesquisa efetuada pelo utilizador, esta janela possui uma lista com os vários resultados do lado esquerdo, enquanto do lado oposto, possui uma imagem da capa do livro, os autores, um breve resumo e um botão que permite aceder ao livro em formato PDF.

```
public searchResults(string _userName, SqlCommand _command)
{
    InitializeComponent();
    userName = _userName;
    lblUser.Content = userName;
    sqlCommand = _command;
    dataGrid.CanUserAddRows = false;
    Bind();
    rowStyle.Setters.Add(new EventSetter(DataGridRow.MouseDoubleClickEvent, new MouseButtonEventHandler(rowDoubleClick)));
    dataGrid.RowStyle = rowStyle;
}
```

Figura 35: Método *searchResults*

O método *searchResults* recebe dois argumentos, a identificação do utilizador e um comando SQL para pesquisa. Este método inicializa todos os componentes da janela, bloqueia a introdução de linhas por parte do utilizador na lista de resultados, cria um *event handler* associado ao duplo clique sobre cada linha e chama o método *Bind*.

```
protected void Bind()
{
    sqlConn.ConnectionString =
Properties.Settings.Default.BibliotecaConnectionString;
    string connectionString = sqlConn.ConnectionString.ToString();
    SqlDataAdapter sqlDataAdapter = new SqlDataAdapter(sqlCommand.CommandText,
connectionString);
    sqlDataAdapter.Fill(dataSet, "livrosTitulo");
    dataTable = dataSet.Tables["livrosTitulo"];
    dataGrid.DataContext = dataTable.DefaultView;
}
```

Figura 36: Método *Bind* (*searchResults*)

O papel deste método é proporcionar o preenchimento da lista de resultados com a informação que corresponda ao parâmetro de pesquisa introduzido pelo utilizador na janela *Home* ou *Advanced Search*. Através da *connectionString* de ligação à base de dados e do comando SQL, é criado um *DataAdapter* que irá preencher uma tabela com o nome “*booksResult*” no *dataSet* com o resultado da pesquisa. Por fim, o resultado da consulta à base de dados é atribuído a uma *DataGrid* que assume a forma de lista de resultados.

```
private void rowDoubleClick(object sender, MouseButtonEventArgs e)
{
    DataGridRow row = sender as DataGridRow;
    if (row.GetIndex() != previousClickedRow)
    {
        dataRow = dataTable.Rows[row.GetIndex()];
        if (dataRow.ItemArray[4] != DBNull.Value)
        {
            string bookCover = "\\\\192.168.1.2\\Biblioteca\\bookImages\\" +
dataRow.ItemArray[4].ToString();
            try
            {
                BitmapImage bitMapImage = new BitmapImage();
                bitMapImage.BeginInit();
                bitMapImage.UriSource = new Uri(bookCover);
```

```

        bitMapImage.CacheOption = BitmapCacheOption.OnLoad;
        imgBookCover.Source = bitMapImage;
        bitMapImage.EndInit();
    }
    catch (Exception error)
    {
        imgBookCover.Source = null;
    }
}
dataTableAutores =
tableAdapterAutores.getDataByAutorISBN(dataRow.ItemArray[0].ToString());
txtClick.Visibility = Visibility.Hidden;
border.Visibility = Visibility.Hidden;
stackAutor.Children.Clear();
foreach (dataSetBiblioteca.AutoresRow tbAutoresRow in
dataTableAutores.Rows)
{
    TextBlock text = new TextBlock();
    text.Text = tbAutoresRow.Nome;
    text.FontSize = 20;
    stackAutor.Children.Add(text);
}
if (dataTableAutores.Rows.Count == 1)
    lblAutor.Content = "Autor:";
else
    lblAutor.Content = "Autores:";
lblAutor.Visibility = Visibility.Visible;
scrollViwer.Visibility = Visibility.Visible;
lblResumo.Visibility = Visibility.Visible;
btnOpenBook.Visibility = Visibility.Visible;
txtResumo.Visibility = Visibility.Visible;
txtResumo.Text = dataRow.ItemArray[3].ToString();
previousClickedRow = row.GetIndex();
}
}

```

Figura 37: Event handler `rowDoubleClick`

O *event handler* *rowDoubleClick* permite que o utilizador efetue um duplo clique com o rato numa das linhas de resultados e consiga visualizar a informação sobre o livro pretendido. Esta informação contém a capa, os autores, o resumo e o ficheiro PDF do livro. Este evento começa por identificar qual a linha que o utilizador fez o duplo clique e efetua um teste para verificar se a informação do livro que o utilizador pretende aceder não está já visível, este teste proporciona uma otimização da aplicação. É verificado se o livro possui capa definida ou não, se possuir é definido o caminho de rede para a localização da capa e a imagem é carregada, sendo sempre limpo a *cache* antes de carregar a imagem para a aplicação, isto permite que caso a capa do livro seja alterada, o utilizador tenha sempre acesso à última versão da capa. O próximo passo é efetuar uma consulta à base de dados para obter os autores do livro, sendo estes carregados num *StackPanel* através de um ciclo

*foreach*. Por fim falta apenas preencher o resumo do livro, e atribuir o número da linha consultada à variável *previousClickedRow* de forma a poder efetuar o primeiro teste de condição explicado em cima.

```
private void openPDF(object sender, RoutedEventArgs e)
{
    if (dataRow.ItemArray[6] == DBNull.Value)
    {
        MessageBox.Show("Não foi possível visualizar o livro.", "Erro",
        MessageBoxButton.OK, MessageBoxImage.Warning);
        return;
    }
    try
    {
        System.Diagnostics.Process process = new System.Diagnostics.Process();
        string path = "\\\\" + 192.168.1.2 + "\\Biblioteca\\Books\\" +
        dataRow.ItemArray[6].ToString();
        Uri pdf = new Uri(path, UriKind.RelativeOrAbsolute);
        process.StartInfo.FileName = pdf.LocalPath;
        process.Start();
        process.WaitForExit();
    }
    catch(Exception error)
    {
        MessageBox.Show("Não foi possível visualizar o livro.", "Erro",
        MessageBoxButton.OK, MessageBoxImage.Warning);
    }
}
```

Figura 38: *Event handler openPDF*

O botão *Open PDF* permite consultar o livro em formato PDF. Para isso está associado ao evento *click* o *event handler openPDF*. Este começa por verificar se a coluna “PDF, que tem o nome do ficheiro do livro, possui informação, caso possua, o livro é aberto no programa de visualização de ficheiro em formato PDF instalado (ex: *Adobe Reader*). Nota ainda para a variável *path* que possui o caminho de rede para a pasta onde estão os livros em formato PDF. De realçar que estes livros estão protegidos contra a cópia e impressão não autorizada.

```
private void HandleCanExecute(object sender, CanExecuteRoutedEventArgs e)
{
    if (e.Command == ApplicationCommands.Cut || e.Command ==
ApplicationCommands.Copy || e.Command == ApplicationCommands.Paste)
    {
        e.CanExecute = false;
        e.Handled = true;
    }
}
```

Figura 39: Event handler *HandleCanExecute*

O evento *HandleCanExecute* está associado ao elemento *TextBox* que possui o resumo do livro. Este restringe a função copiar, de forma a impedir que o utilizador copie o resumo do livro através da aplicação.

```
private void Image_MouseLeftButtonDown_1(object sender, MouseButtonEventArgs e)
{
    Home home = new Home(userName);
    this.Close();
    home.Show();
}
```

Figura 40: Event handler *Image\_MouseLeftButtonDown\_1*

O utilizador ao clicar no botão retroceder regressa à janela *Home* e fecha a janela atual, é esta a função do *event handler* na figura 40. Este evento é usado em todos os botões de retroceder.

### 3.3.7. Pessoas

Regressando à janela *Home*, vamos lançar um olhar sobre a componente de gestão e administração da aplicação.

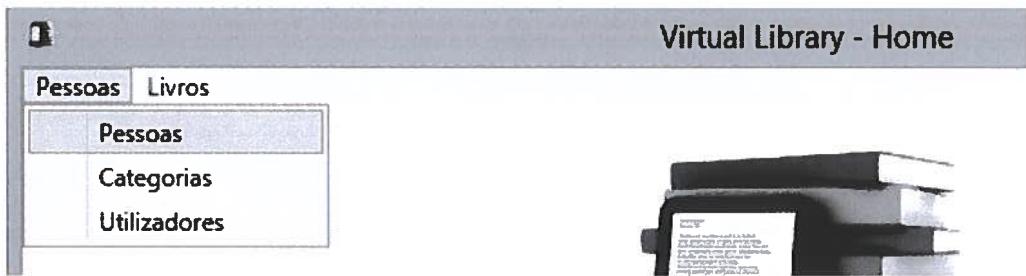


Figura 41: Menu Pessoas

O menu pessoas permite a administração dos colaboradores, utilizadores e tipos de utilizadores da aplicação. Esta informação está nas tabelas TiposPessoas, Pessoas, Emails, Telefones e Utilizadores.

IdPessoa	Categoria Pessoa	Nome	Cartão Cidadão
1	Informática	Carlos Jorge Tregeira Sota	130 [REDACTED]

Morada	Andar	Letra	Código Postal	Localidade	País
Rua de [REDACTED]	1	[REDACTED]	[REDACTED]-195	Montemor [REDACTED]	Portugal

Email:

Telefone / Telemovel:

**Novo**

**Gravar**

**Eliminar**

1 de 2

Figura 42: Janela Pessoas

A janela Pessoas permite a gestão de quatro tabelas distintas, sendo estas, as tabelas TiposPessoa, Pessoa, Emails e Telefone. Possui cinco botões de navegação que permitem avançar para o registo seguinte, retroceder, ir para o último registo, para o primeiro registo e regressar à janela anterior. Existe ainda quatro botões, apesar de apenas três serem visíveis, são estes, o botão “Novo”, “Gravar”, “Eliminar” e “Cancelar”. O botão “Cancelar” apenas é visível após o utilizador clicar no botão “Novo”. Por último, temos os botões que permitem

adicionar ou remover os endereços email e os números de contato da pessoa. É importante realçar ainda que a caixa de texto “IdPessoa” está bloqueada para edição, o valor deste campo representa a chave primária da tabela Pessoas sendo incrementado automaticamente sempre que um novo registo é feito.

```
protected void Bind()
{
    i = 0;
    dataTablePessoas = tableAdapterPessoas.getPessoas();
    tbPessoasRow = (dataSetBiblioteca.PessoasRow)dataTablePessoas.Rows[0];
    lblReg.Content = (i + 1).ToString() + " de " +
    dataTablePessoas.Count().ToString();
    txtIdPessoa.Text = tbPessoasRow.idPessoa.ToString();
    fillcbPessoa();
    fillListView();
    cbPessoa.SelectedValue = tbPessoasRow.Tipo.ToString();
    if(tbPessoasRow.IsNomeNull() == false)
        txtNome.Text = tbPessoasRow.Nome.ToString();
    if(tbPessoasRow.IsCCNull() == false)
        txtCC.Text = tbPessoasRow.CC.ToString();
    if(tbPessoasRow.IsRuaNull() == false)
        txtRua.Text = tbPessoasRow.Rua.ToString();
    if(tbPessoasRow.IsAndarNull() == false)
        txtAndar.Text = tbPessoasRow.Andar.ToString();
    if (tbPessoasRow.IsLetraNull() == false)
        txtLetra.Text = tbPessoasRow.Letra.ToString();
    if(tbPessoasRow.IscodPostalNull() == false)
        txtCodPostal.Text = tbPessoasRow.codPostal.ToString();
    if(tbPessoasRow.IsLocalidadeNull() == false)
        txtLocalidade.Text = tbPessoasRow.Localidade.ToString();
    if(tbPessoasRow.IsPaisNull() == false)
        txtPais.Text = tbPessoasRow.Pais.ToString();
}
```

Figura 43: Método *Bind* (*Pessoas*)

O método *Bind*, já utilizado em janelas anteriores, é utilizado para atribuir os valores da base de dados aos campos presentes na janela *Pessoas*. Este método é utilizado ao longo do restante código para garantir que os dados que o utilizador está a visualizar estão atualizados.

```

protected void fillcbPessoa()
{
    dataTableTiposPessoa = tableAdapterTiposPessoa.getTipoPessoa();
    cbPessoa.ItemsSource = dataTableTiposPessoa;
    cbPessoa.DisplayMember = "Desig";
    cbPessoa.SelectedValuePath = "idTipo";
}

protected void fillListView()
{
    dataTableEmails =
    tableAdapterEmails.getDataByIdPessoa(tbPessoasRow.idPessoa);
    dataTableTelefone =
    tableAdapterTelefone.getDataByIdPessoa(tbPessoasRow.idPessoa);
    lvEmail.DataContext = dataTableEmails.DefaultView;
    lvContact.DataContext = dataTableTelefone.DefaultView;
}

```

Figura 44: Métodos *fillcbPessoa* e *fillListView*

Os métodos *fillcbPessoa* e *fillListView* são utilizados no método *Bind* e preenchem a *ComboBox* “Categoria de Pessoas” e as *ListView*’s “Emails” e “Telefones” respetivamente.

```

protected void clearValues()
{
    txtNome.Clear();
    txtCC.Clear();
    txtRua.Clear();
    txtAndar.Clear();
    txtLetra.Clear();
    txtCodPostal.Clear();
    txtLocalidade.Clear();
    txtPais.Clear();
}

```

Figura 45: Método *clearValues*

O método *clearValues* é utilizado durante a execução do código do método *Bind*, e permite limpar as caixas e texto da janela antes de os preencher com nova informação.

```

private void btnNext_Click_1(object sender, RoutedEventArgs e)
{
    if (i >= dataTablePessoas.Count() - 1 || value == true)
        return;
    i++;
    tbPessoasRow = (dataSetBiblioteca.PessoasRow)dataTablePessoas.Rows[i];
    lblReg.Content = (i + 1).ToString() + " de " +
    dataTablePessoas.Count().ToString();
    txtIdPessoa.Text = tbPessoasRow.idPessoa.ToString();
    cbPessoa.SelectedValue = tbPessoasRow.Tipo.ToString();
    fillListView();
    clearValues();
    if (tbPessoasRow.IsNomeNull() == false)
        txtNome.Text = tbPessoasRow.Nome.ToString();
    if (tbPessoasRow.IsCCNull() == false)
        txtCC.Text = tbPessoasRow.CC.ToString();
    if (tbPessoasRow.IsRuaNull() == false)
        txtRua.Text = tbPessoasRow.Rua.ToString();
    if (tbPessoasRow.IsAndarNull() == false)
        txtAndar.Text = tbPessoasRow.Andar.ToString();
    if (tbPessoasRow.IsLetraNull() == false)
        txtLetra.Text = tbPessoasRow.Letra.ToString();
    if (tbPessoasRow.IscodPostalNull() == false)
        txtCodPostal.Text = tbPessoasRow.codPostal.ToString();
    if (tbPessoasRow.IsLocalidadeNull() == false)
        txtLocalidade.Text = tbPessoasRow.Localidade.ToString();
    if (tbPessoasRow.IsPaisNull() == false)
        txtPais.Text = tbPessoasRow.Pais.ToString();
}

```

Figura 46: Event Handler *btnNext\_Click\_1*

Quando o utilizador clica no botão para avançar para o registo seguinte é executado o *event handler* da figura 46. Este evento verifica se a variável *i*, que controla o número de avanços e retrocessos na navegação entre os registos, não excede o número de linhas da tabela Pessoas, assim se o utilizador estiver a visualizar o último registo da tabela não pode avançar mais. O restante código é bastante semelhante ao código do método *Bind*. Os eventos *btnPrevious\_Click\_1*, *btnLast\_Click\_1* e *btnFirst\_Click\_1* são executados quando o utilizador retrocede nos registos, avança para o último registo e para o primeiro registo, respetivamente. Estes eventos possuem um código muito semelhante ao descrito na figura 46, a principal diferença reside no teste que é feito à variável *i*. Assim sendo, quando o evento *btnPrevious\_Click\_1* é executado é verificado se a variável *i* não é igual ou inferior 0 ou se a variável booleana *value* não é igual a verdadeiro. A variável *value* é uma variável global e é iniciada a falso. Esta passa a verdadeiro se ocorrer um clique sobre o botão ‘Novo’. Se uma das condições anteriores verificar-se, o utilizador já está no primeiro registo ou encontra-se no modo de inserir um novo registo. O evento *btnLast\_Click\_1* testa apenas se o *value* é

verdadeiro, caso esta condição se verifique, o utilizador encontra-se, como referido anteriormente, em modo de edição. Neste evento o valor da variável *i* é sempre o valor do número total de linhas da tabela Pessoas. Por último, o evento *btnFirst\_Click\_1* avança para o primeiro registo e testa apenas se o valor da variável *value* é verdadeiro, caso não seja, atribui o valor 0 à variável *i* e mostra a linha 0 da tabela Pessoas ao utilizador.

```
private void insertDataClick(object sender, RoutedEventArgs e)
{
    lblMsg.Visibility = Visibility.Visible;
    btnEliminar.Visibility = Visibility.Hidden;
    btnCancelar.Visibility = Visibility.Visible;
    lblEmail.Visibility = Visibility.Hidden;
    lblContact.Visibility = Visibility.Hidden;
    txtEmail.Visibility = Visibility.Hidden;
    txtContact.Visibility = Visibility.Hidden;
    clearValues();
    cbPessoa.SelectedIndex = -1;
    txtIdPessoa.Clear();
    lvEmail.DataContext = null;
    lvContact.DataContext = null;
    value = true;
}

private void cancelClick(object sender, RoutedEventArgs e)
{
    lblMsg.Visibility = Visibility.Hidden;
    btnEliminar.Visibility = Visibility.Visible;
    btnCancelar.Visibility = Visibility.Hidden;
    lblEmail.Visibility = Visibility.Visible;
    lblContact.Visibility = Visibility.Visible;
    txtEmail.Visibility = Visibility.Visible;
    txtContact.Visibility = Visibility.Visible;
    Bind();
    value = false;
}
```

Figura 47: Event Handler *insertDataClick* e *cancelClick*

O evento *insertDataClick* é executado quando o utilizador clica no botão “Novo” para inserir um novo registo. O código deste evento vai apresentar uma mensagem ao utilizador a informar que este se encontra em modo de edição, torna visível o botão “Cancelar” e esconde o botão “Eliminar”, da mesma forma, remove provisoriamente a possibilidade de adicionar email's e contatos enquanto este registo não for gravado e coloca a variável *value* com valor verdadeiro. Por sua vez o evento *cancelClick* é executado quando o utilizador clica sobre o

botão “Cancelar”. Este evento volta a colocar visível o botão “Eliminar”, a possibilidade de adicionar email's e contatos, executa o método *Bind* e atribui o valor falso à variável *value*.

```
private void saveDataClick(object sender, RoutedEventArgs e)
{
    if (cbPessoa.SelectedIndex > -1 && txtNome.Text.Length > 0 &&
        txtCC.Text.Length > 0 && txtRua.Text.Length > 0 && txtCodPostal.Text.Length > 0 &&
        txtLocalidade.Text.Length > 0 && txtPais.Text.Length > 0)
    {
        tbPessoasRow = (dataSetBiblioteca.PessoasRow)dataTablePessoas.Rows[i];
        string idPessoa = txtIdPessoa.Text.Replace(" ", String.Empty);
        string tipoUtilizador = cbPessoa.SelectedValue.ToString();
        string nome = txtNome.Text.Trim();
        string cc = txtCC.Text.Replace(" ", String.Empty);
        string rua = txtRua.Text.Trim();
        string andar = txtAndar.Text.Replace(" ", String.Empty);
        string letra = txtLetra.Text.Replace(" ", String.Empty);
        string codPostal = txtCodPostal.Text.Replace(" ", String.Empty);
        string localidade = txtLocalidade.Text.Replace(" ", String.Empty);
        string pais = txtPais.Text.Replace(" ", String.Empty);
        if (value == true)
        {
            try
            {
                if (andar == "")
                    if (letra == "")
                        tableAdapterPessoas.Insert(tipoUtilizador,
Convert.ToInt32(cc), nome, rua, null, null, null, localidade, codPostal, pais);
                    else
                        tableAdapterPessoas.Insert(tipoUtilizador,
Convert.ToInt32(cc), nome, rua, null, null, letra, localidade, codPostal, pais);
                    else
                        tableAdapterPessoas.Insert(tipoUtilizador,
Convert.ToInt32(cc), nome, rua, null, Convert.ToInt32(andar), letra, localidade,
codPostal, pais);
            }
            catch (Exception error)
            {
                MessageBox.Show(error.Message.ToString(), "Erro",
MessageBoxButton.OK, MessageBoxIcon.Error);
                return;
            }
            MessageBox.Show("Novo registo inserido com sucesso", "Confirmação",
MessageBoxButton.OK, MessageBoxIcon.Information);
            lblMsg.Visibility = Visibility.Hidden;
            btnCancelar.Visibility = Visibility.Hidden;
            btnEliminar.Visibility = Visibility.Visible;
            lblEmail.Visibility = Visibility.Visible;
            lblContact.Visibility = Visibility.Visible;
            txtEmail.Visibility = Visibility.Visible;
            txtContact.Visibility = Visibility.Visible;
            Bind();
            value = false;
        }
    }
}
```

```

        else
        {
            try
            {
                tbPessoasRow.Tipo = tipoUtilizador;
                tbPessoasRow.Nome = nome;
                tbPessoasRow.CC = Convert.ToInt32(cc);
                tbPessoasRow.Rua = rua;
                if(andar == "")
                    tbPessoasRow.SetAndarNull();
                else
                    tbPessoasRow.Andar = Convert.ToInt32(andar);
                if (letra == "")
                    tbPessoasRow.SetLetraNull();
                else
                    tbPessoasRow.Letra = letra;
                tbPessoasRow.codPostal = codPostal;
                tbPessoasRow.Localidade = localidade;
                tbPessoasRow.Pais = pais;
                tableAdapterPessoas.Update(dataTablePessoas);
            }
            catch (Exception error)
            {
                MessageBox.Show(error.Message.ToString(), "Erro",
                MessageBoxButtons.OK, MessageBoxIcon.Error);
                return;
            }
            MessageBox.Show("Dados gravados com sucesso!", "Confirmação",
            MessageBoxButtons.OK, MessageBoxIcon.Information);
            Bind();
        }
        else
        {
            MessageBox.Show("Não é possível inserir registo nulos. Campos
obrigatórios: Categoria de Pessoa, Nome, Cartão Cidadão, Morada, Código de Postal,
Localidade e País", "Erro", MessageBoxButtons.OK, MessageBoxIcon.Error);
        }
    }
}

```

Figura 48: Event Handler *saveDataClick*

O evento *saveDataClick* guarda as alterações feitas nos registos atuais ou nos novos registos. Este evento analisa numa fase inicial se os campos de preenchimento obrigatório estão preenchidos, caso estejam os valores desses campos são guardados em variáveis criadas para esse efeito, mas com a particularidade da utilização da propriedade *Trim* e *Replace* para eliminar os espaços em branco que possam existir por lapso do utilizador. Como vimos anteriormente, se o valor da variável *value* for verdadeiro, significa que o utilizador está a inserir um novo registo, logo este evento utiliza essa variável para perceber se tem de inserir um novo registo ou atualizar um existente. Destaque para o facto dos campos “andar” e

“letra” aceitarem valores *null*, assim é necessário três diferentes *insert's*. Se o valor da variável *value* for falso, a aplicação atualiza o registo com a nova informação introduzida. O utilizador é sempre notificado através de uma caixa de diálogo do sucesso da inserção de um novo registo ou da atualização, da mesma forma, se algo estiver incorreto o utilizador é notificado com uma caixa de diálogo. Por último o método *Bind* é sempre executado para atualizar os registas no ecrã a que utilizador tem acesso.

```
private void deleteClick(object sender, RoutedEventArgs e)
{
    UtilizadoresTableAdapter tableAdapterUtilizadores = new
    UtilizadoresTableAdapter();
    dataSetBiblioteca.UtilizadoresDataTable dataTableUtilizadores;
    dataSetBiblioteca.UtilizadoresRow tbUtilizadoresRow;
    dataTableUtilizadores =
        tableAdapterUtilizadores.getByName(lblUser.Content.ToString());
    tbUtilizadoresRow =
        (dataSetBiblioteca.UtilizadoresRow)dataTableUtilizadores[0];
    if (tbUtilizadoresRow.idPessoa == tbPessoasRow.idPessoa)
    {
        MessageBox.Show("Não pode eliminar um utilizador com sessão iniciada.",
        "Erro", MessageBoxButton.OK, MessageBoxIcon.Warning);
        return;
    }
    MessageBoxResult confirm = MessageBox.Show("Tem a certeza que pretende
    eliminar este registo?", "Atenção", MessageBoxButton.YesNo,
    MessageBoxIcon.Question);
    if (confirm == MessageBoxResult.Yes)
    {
        try
        {
            tbPessoasRow =
                (dataSetBiblioteca.PessoasRow)dataTablePessoas.Rows[i];
            tbPessoasRow.Delete();
            tableAdapterPessoas.Update(dataTablePessoas);
        }
        catch (Exception error)
        {
            MessageBox.Show(error.Message.ToString(), "Erro",
            MessageBoxButton.OK, MessageBoxIcon.Warning);
            return;
        }
        MessageBox.Show("Registo eliminado", "Aviso", MessageBoxButton.OK,
        MessageBoxIcon.Information);
        Bind();
    }
}
```

Figura 49: Event Handler *deleteClick*

O evento *deleteClick* está associado ao botão “Eliminar” e tem propósito eliminar o registo que o utilizador está a visualizar. Contudo, existe uma condição a verificar antes de eliminar o registo. A primeira condição verifica se o registo a eliminar não está associado ao utilizador que tem a sessão iniciada. Caso se verifique esta condição, o utilizador é alertado para o efeito. Ultrapassada esta condição, surge uma caixa de diálogo que solicita ao utilizador a confirmação da eliminação deste registo. Por último, e mais uma vez, o método *Bind* é executado.

```
private void addEmail(object sender, RoutedEventArgs e)
{
    if (txtEmail.Text.Length == 0) return;
    string idPessoa = txtIdPessoa.Text.Replace(" ", String.Empty);
    string email = txtEmail.Text.Replace(" ", String.Empty);
    try
    {
        tableAdapterEmails.Insert(Convert.ToInt32(idPessoa), email);
    }
    catch (Exception error)
    {
        MessageBox.Show(error.Message.ToString(), "Erro", MessageBoxButton.OK,
MessageBoxImage.Error);
        return;
    }
    MessageBox.Show("Novo email inserido com sucesso", "Confirmação",
MessageBoxButton.OK, MessageBoxIcon.Information);
    txtEmail.Clear();
    Bind();
}

private void addContact(object sender, RoutedEventArgs e)
{
    if (txtContact.Text.Length == 0) return;
    string idPessoa = txtIdPessoa.Text.Replace(" ", String.Empty);
    string contact = txtContact.Text.Replace(" ", String.Empty);
    try
    {
        tableAdapterTelefone.Insert(Convert.ToInt32(idPessoa), contact);
    }
    catch (Exception error)
    {
        MessageBox.Show(error.Message.ToString(), "Erro", MessageBoxButton.OK,
MessageBoxImage.Error);
        return;
    }
    MessageBox.Show("Novo contacto inserido com sucesso", "Confirmação",
MessageBoxButton.OK, MessageBoxIcon.Information);
    txtContact.Clear();
    Bind();
}
```

Figura 50: Event Handler's *addEmail* e *addContact*

A figura 50 representa os eventos *addEmail* e *addContact* que são executados quando o utilizador insere um novo email ou contacto. Dada a similaridade entre os dois eventos, estes estão representados numa única figura. Para adicionar um novo email ou contacto, o utilizador deve preencher a caixa de texto para o efeito e clicar no botão. Ambos os eventos verificam se as caixas de texto estão preenchidas, caso estejam é utilizado o valor do “IdPessoa”<sup>13</sup>, e é removido os espaços no correio eletrónico ou contacto. O passo seguinte é efetuar um *insert* na tabela correspondente, Emails ou Telefones, e confirmar o sucesso da operação ao utilizador através de uma caixa de diálogo. Por último é executado o método *Bind* e é removido o valor da caixa de texto “Email” ou “Telem.”.

```
private void removeEmail(object sender, RoutedEventArgs e)
{
    if (lvEmail.SelectedItems.Count > 0)
    {
        string idPessoa = txtIdPessoa.Text.Replace(" ", String.Empty);
        string email = txtEmail.Text.Replace(" ", String.Empty);
        try
        {
            tableAdapterEmails.Delete(Convert.ToInt32(idPessoa), email);
        }
        catch (Exception error)
        {
            MessageBox.Show(error.Message.ToString(), "Erro",
                MessageBoxButton.OK, MessageBoxIcon.Error);
            return;
        }
        MessageBox.Show("Email removido com sucesso", "Confirmação",
            MessageBoxButton.OK, MessageBoxIcon.Information);
        txtEmail.Clear();
        Bind();
    }
}

private void removeContact(object sender, RoutedEventArgs e)
{
    if (lvContact.SelectedItems.Count > 0)
    {
        string idPessoa = txtIdPessoa.Text.Replace(" ", String.Empty);
        string contact = txtContact.Text.Replace(" ", String.Empty);
        try
        {
            tableAdapterTelefone.Delete(Convert.ToInt32(idPessoa), contact);
        }
        catch (Exception error)
        {
```

---

<sup>13</sup> Chave primária da tabela Pessoas

```

        MessageBox.Show(error.Message.ToString(), "Erro",
        MessageBoxButtons.OK, MessageBoxIcon.Error);
        return;
    }
    MessageBox.Show("Contacto removido com sucesso", "Confirmação",
    MessageBoxButtons.OK, MessageBoxIcon.Information);
    txtContact.Clear();
    Bind();
}
}

```

Figura 51: Event Handler's removeEmail e removeContact

Da mesma forma que figura 50 representa dois eventos a figura 51 representa os eventos *removeEmail* e *removeContact*, devido à sua similaridade. Estes são executados quando o utilizador remove um endereço de email ou contacto. Para remover um endereço email ou contacto, o utilizador deve clicar no valor que pretende eliminar e clicar no botão “-”. Ambos os eventos verificam se existe um valor selecionado, caso exista, é utilizado o valor do “IdPessoa”<sup>14</sup> e o valor do email ou contacto para eliminar o registo correto. O sucesso da operação é confirmado ao utilizador através de uma caixa de diálogo. Por último é executado o método *Bind*.

### 3.3.8. Utilizadores



Figura 52: Janela Utilizadores

<sup>14</sup> Chave primária da tabela Pessoas

A janela Utilizadores permite a gestão dos utilizadores da aplicação. Tal como a janela anterior, esta é constituída por cinco botões de navegação, primeiro registo, registo anterior, registo seguinte, último registo e regressar à janela anterior. Possui ainda quatro botões que permitem adicionar um novo registo, gravar as alterações, eliminar o registo atual e cancelar.

O código por detrás desta janela possui métodos e eventos idênticos à janela Pessoas que foi abordado no ponto 3.3.7. O método *Bind* é executado quando a janela é aberta e efetua uma pesquisa na base de dados e atribui aos campos da janela os valores que estão guardados na tabela Utilizadores.

Os eventos *btnNext\_Click\_1*, *btnPrevious\_Click\_1*, *btnLast\_Click\_1* e *btnFirst\_Click\_1* são executados durante a navegação pelos registos, estes permitem avançar, retroceder, ir para o último ou para o primeiro registo. Estes eventos controlam o valor das variáveis globais *i* e *value*, sendo a variável *value* é iniciada a falso, se o seu valor for verdadeiro, indica que o utilizador clicou no botão ‘Novo’ e está a inserir um novo registo. Desta forma os botões de navegação dos registos estão desativados até o utilizador permanecer neste modo. A variável *i* é utilizada para controlar o número de registos visualizados pelo utilizador, não permitindo que o utilizador avance para um número de registo que excede o número de linhas da tabela.

```
private void btnPrevious_Click_1(object sender, RoutedEventArgs e)
{
    if (i <= 0 || value == true)
        return;
    i--;
    tbUtilizadoresRow =
(dataSetBiblioteca.UtilizadoresRow)dataTableUtilizadores.Rows[i];
    lblReg.Content = (i + 1).ToString() + " de " +
dataTableUtilizadores.Count().ToString();
    txtIdPessoa.Text = tbUtilizadoresRow.idPessoa.ToString();
    txtIdUtilizador.Text = tbUtilizadoresRow.idUtilizador.ToString();
    cbTipoUtilizador.SelectedValue = tbUtilizadoresRow.tipoUtilizador.ToString();
    txtUsername.Text = tbUtilizadoresRow.userName.ToString();
    txtPassword.Password = tbUtilizadoresRow.userPassword.ToString();
}
```

Figura 53: Event Handler *btnPrevious\_Click\_1*

```

private void saveDataClick(object sender, RoutedEventArgs e)
{
    if (txtIdPessoa.Text.Length > 0 || txtIdUtilizador.Text.Length > 0 || 
    txtPassword.Password.Length > 0 || txtUsername.Text.Length > 0 || 
    cbTipoUtilizador.SelectedIndex > -1)
    {
        string idPessoa = txtIdPessoa.Text.Replace(" ", String.Empty);
        string idUtilizador = txtIdUtilizador.Text.Replace(" ", String.Empty);
        string utilizador = txtUsername.Text.Replace(" ", String.Empty);
        string tipoUtilizador = cbTipoUtilizador.SelectedValue.ToString();
        string password = txtPassword.Password.Replace(" ", String.Empty);
        if (value == true)
        {
            try
            {
                tableAdapterUtilizadores.Insert(Convert.ToInt32(idPessoa),
                Convert.ToInt32(idUtilizador), utilizador,
                cbTipoUtilizador.SelectedValue.ToString(), Cripto.sha256(password));
            }
            catch (Exception error)
            {
                MessageBox.Show(error.Message.ToString(), "Erro",
                MessageBoxButton.OK, MessageBoxImage.Error);
                return;
            }
            MessageBox.Show("Novo registo inserido com sucesso", "Confirmação",
            MessageBoxButton.OK, MessageBoxImage.Information);
            lblMsg.Visibility = Visibility.Hidden;
            btnCancelar.Visibility = Visibility.Hidden;
            txtIdPessoa.IsReadOnly = true;
            txtIdUtilizador.IsReadOnly = true;
            Bind();
            value = false;
        }
        else
        {
            tbUtilizadoresRow =
            (dataSetBiblioteca.UtilizadoresRow)dataTableUtilizadores.Rows[i];
            if (idPessoa != tbUtilizadoresRow.idPessoa.ToString() ||
            idUtilizador != tbUtilizadoresRow.idUtilizador.ToString() || utilizador !=
            tbUtilizadoresRow.userName.ToString() || tipoUtilizador !=
            tbUtilizadoresRow.tipoUtilizador.ToString() || password !=
            tbUtilizadoresRow.userPassword.ToString())
            {
                try
                {
                    dataTablePessas =
                    tableAdapterPessoas.getDataByIdPessoa(tbUtilizadoresRow.idPessoa);
                    tbPessoasRow =
                    (dataSetBiblioteca.PessoasRow)dataTablePessas.Rows[0];
                    tbUtilizadoresRow.userName = utilizador;
                    tbUtilizadoresRow.userPassword = Cripto.sha256(password);
                    tbUtilizadoresRow.tipoUtilizador = tipoUtilizador;
                    tbPessoasRow.Tipo = tipoUtilizador;
                    tableAdapterUtilizadores.Update(dataTableUtilizadores);
                    tableAdapterPessoas.Update(dataTablePessas);
                }
                catch (Exception error)
                {
                    MessageBox.Show(error.Message.ToString(), "Erro",
                    MessageBoxButton.OK, MessageBoxImage.Error);
                    return;
                }
            }
        }
    }
}

```

```

        MessageBox.Show("Dados gravados com sucesso!", "Confirmação",
    MessageBoxButtons.OK, MessageBoxIcon.Information);
    Bind();
}
}
else
{
    MessageBox.Show("Não é possível inserir registo nulos", "Erro",
    MessageBoxButtons.OK, MessageBoxIcon.Error);
}
}

```

Figura 54: Event Handler *saveDataClick* (Utilizadores)

O evento *saveDataClick* guarda as alterações feitas nos registos atuais e nos novos registos. Este evento analisa numa fase inicial se os campos de preenchimento obrigatório estão preenchidos. Caso estejam esses valores são guardados em variáveis criadas para o efeito com a particularidade da utilização da propriedade *Trim* e *Replace* para eliminar os espaços em branco que possam existir por lapso do utilizador. Como vimos anteriormente, se o valor da variável *value* for verdadeiro, significa que o utilizador está a inserir um novo registo, logo este evento utiliza essa variável para perceber se tem de inserir um novo registo ou atualizar um existente. O utilizador é sempre notificado através de uma caixa de diálogo do sucesso da inserção de um novo registo ou da atualização, da mesma forma, se algo incorreto ocorrer o utilizador é notificado com uma caixa de diálogo. Destaque ainda que no caso da atualização de dado, o evento atualiza o valor do campo “Categoria Utilizador” nas tabelas Utilizadores e Pessoas. Por último o método *Bind* é sempre executado para atualizar os registos no ecrã a que utilizador tem acesso.

```

private void deleteClick(object sender, RoutedEventArgs e)
{
    if (txtUsername.Text.ToString() == lblUser.Content.ToString())
    {
        MessageBox.Show("Não pode eliminar um utilizador com sessão iniciada.",
    "Erro", MessageBoxButtons.OK, MessageBoxIcon.Warning);
        return;
    }
    MessageBoxResult confirm = MessageBox.Show("Tem a certeza que pretende
eliminar este registo?", "Atenção", MessageBoxButtons.YesNo,
MessageBoxImage.Question);

```

```

    if (confirm == MessageBoxResult.Yes)
    {
        try
        {
            tbUtilizadoresRow =
(dataSetBiblioteca.UtilizadoresRow)dataTableUtilizadores.Rows[i];
            tbUtilizadoresRow.Delete();
            tableAdapterUtilizadores.Update(dataTableUtilizadores);
        }
        catch (Exception error)
        {
            MessageBox.Show(error.Message.ToString(), "Erro",
MessageBoxButton.OK, MessageBoxIcon.Warning);
            return;
        }
        MessageBox.Show("Registo eliminado", "Aviso", MessageBoxButton.OK,
MessageBoxImage.Information);
        Bind();
    }
}

```

Figura 55: Event Handler *deleteClick* (*Utilizadores*)

O evento da figura 55, *deleteClick*, está associado ao botão “Eliminar” e tem a função de eliminar o registo que o utilizador esteja a visualizar. Contudo, existe uma condição que verifica se o registo a eliminar não está associado ao utilizador que tem a sessão iniciada. Caso se verifique esta condição, o utilizador é alertado para o efeito. Uma vez ultrapassado esta questão, surge um caixa de diálogo que solicita ao utilizador a confirmação da eliminação deste registo. Por último, e mais uma vez, o método *Bind* é executado.

### 3.3.9. Categorias Pessoas



A janela Categorias de Pessoas permite a gestão das categorias de utilizadores e pessoas da aplicação. É constituída por cinco botões de navegação, primeiro registo, registo anterior, registo seguinte, último registo e regressar à janela anterior. Possui ainda quatro botões que permitem adicionar um novo registo, gravar as alterações, eliminar o registo atual e cancelar.

Seguramente verificou a evidente similaridade nos métodos e eventos entre a janela Pessoas e Utilizadores, desta forma esta janela não foge à regra. Assim apenas será abordado os métodos de inserir, atualizar e eliminar informação, sendo sempre possível consultar todos os eventos e métodos desta janela nos anexos deste projeto.

```
private void saveDataClick(object sender, RoutedEventArgs e)
{
    if (txtIdTipo.Text.Length > 0 || txtDesignacao.Text.Length > 0)
    {
        string idTipo = txtIdTipo.Text.Replace(" ", String.Empty);
        string desig = txtDesignacao.Text.Replace(" ", String.Empty);
        if (value == true)
        {
            try
            {
                tableAdapterTiposPessoa.Insert(idTipo, desig);
            }
            catch (Exception error)
            {

```

```

        MessageBox.Show(error.Message.ToString(), "Erro",
MessageBoxButton.OK, MessageBoxIcon.Error);
        return;
    }
    MessageBox.Show("Novo registo inserido com sucesso", "Confirmação",
MessageBoxButton.OK, MessageBoxIcon.Information);
    lblMsg.Visibility = Visibility.Hidden;
    btnEliminar.Visibility = Visibility.Visible;
    btnCancelar.Visibility = Visibility.Hidden;
    Bind();
    value = false;
}
else
{
    tbTiposPessoaRow =
(dataSetBiblioteca.TiposPessoaRow)dataTableTiposPessoa.Rows[i];
    if (desig != tbTiposPessoaRow.Desig.ToString() || idTipo !=
tbTiposPessoaRow.idTipo.ToString())
    {
        try
        {
            tbTiposPessoaRow.idTipo = idTipo;
            tbTiposPessoaRow.Desig = desig;
            tableAdapterTiposPessoa.Update(dataTableTiposPessoa);
        }
        catch (Exception error)
        {
            MessageBox.Show(error.Message.ToString(), "Erro");
            return;
        }
        MessageBox.Show("Dados gravados com sucesso!", "Confirmação",
MessageBoxButton.OK, MessageBoxIcon.Information);
        Bind();
    }
}
else
{
    MessageBox.Show("Não é possível inserir registos nulos", "Erro",
MessageBoxButton.OK, MessageBoxIcon.Error);
}
}

```

Figura 56: Event Handler *saveDataClick* (Categorias de Pessoas)

O evento *saveDataClick* guarda as alterações feitas nos registos atuais e nos novos registos. Este evento analisa numa fase inicial se os campos de preenchimento obrigatório estão preenchidos, caso estejam, os valores desses campos são guardados em variáveis criadas para o efeito com a particularidade da utilização a propriedade *Trim* e *Replace* para eliminar os espaços em branco que possam existir por lapso do utilizador. Como vimos anteriormente, se o valor da variável *value* for verdadeiro, significa que o utilizador está a inserir um novo registo, logo este evento utiliza essa variável para perceber se tem de inserir

um novo registo ou atualizar um existente. O utilizador é sempre notificado através de uma caixa de diálogo do sucesso da inserção ou da atualização de um registo, da mesma forma, se algo estiver incorreto o utilizador também é notificado. Por último o método *Bind* é sempre executado para atualizar os regtos no ecrã a que utilizador tem acesso.

```
private void deleteClick(object sender, RoutedEventArgs e)
{
    UtilizadoresTableAdapter tableAdapterUtilizadores = new
    UtilizadoresTableAdapter();
    if
    (tableAdapterUtilizadores.getDataByTipoUtilizador(txtIdTipo.Text.ToString()).Count
     > 0)
    {
        MessageBox.Show("Esta categoria está associada a utilizadores
existentes.", "Erro", MessageBoxButton.OK, MessageBoxImage.Warning);
        return;
    }
    MessageBoxResult confirm = MessageBox.Show("Tem a certeza que pretende
eliminar este registo?", "Atenção", MessageBoxButton.YesNo,
MessageBoxImage.Question);
    if (confirm == MessageBoxResult.Yes)
    {
        try
        {
            tbTiposPessoaRow =
            (dataSetBiblioteca.TiposPessoaRow)dataTableTiposPessoa.Rows[i];
            tbTiposPessoaRow.Delete();
            tableAdapterTiposPessoa.Update(dataTableTiposPessoa);
        }
        catch (Exception error)
        {
            MessageBox.Show(error.Message.ToString(), "Erro",
MessageBoxButton.OK, MessageBoxImage.Warning);
            return;
        }
        MessageBox.Show("Registo eliminado", "Aviso", MessageBoxButton.OK,
MessageBoxImage.Information);
        Bind();
    }
}
```

Figura 57: Event Handler *deleteClick* (Categorias de Pessoas)

O evento *deleteClick* está associado ao botão “Eliminar” e tem como função eliminar o registo que o utilizador está a visualizar. Contudo, existe uma condição que verifica se o registo a eliminar não está associado ao registo de nenhum utilizador, ou seja, não é possível eliminar uma categoria que esteve atribuída a um utilizador, sendo necessário remover essa

associação. Caso se verifique esta condição, o utilizador é alertado para o efeito. Ultrapassada esta condição, surge um caixa de diálogo que solicita ao utilizador a confirmação da eliminação deste registo. Por último, e mais uma vez, o método *Bind* é executado.

As três janelas revistas nos pontos 3.3.7, 3.3.8 e 3.3.9 permitem a gestão e administração das tabelas TiposPessoa, Pessoas, Emails, Telefones e Utilizadores.

### 3.3.10. Livros

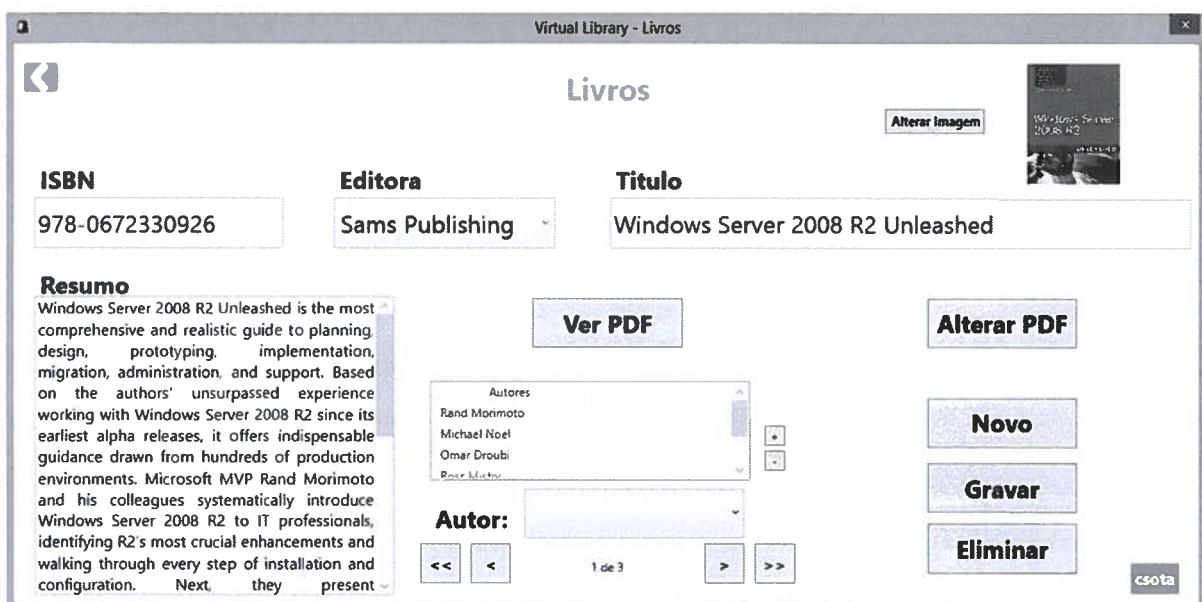


Figura 58: Janela Livros

A janela Livros é possivelmente uma das janelas mais importantes da aplicação. Esta janela permite adicionar, remover e atualizar um livro. É também possível a alteração da capa, do ficheiro PDF com o conteúdo do livro, e dos autores. Verificamos mais uma vez existência de cinco botões de navegação revistos anteriormente em outras janelas da aplicação, bem como os quatro botões que permitem adicionar, remover, gravar e cancelar. O sistema de associação dos autores aos livros é um sistema idêntico ao utilizado na janela Pessoas no ponto 3.3.7, existindo para tal um botão para adicionar e remover um autor. Esta janela possui ainda mais três botões, um que permite alterar a imagem da capa do livro, outro para alterar o ficheiro PDF e um último para consultar o ficheiro PDF. Mais uma vez, existem

métodos e eventos que se repetem nesta janela e que já foi revisto o seu funcionamento em pontos anteriores, pelo que, o foco será dado aos métodos e eventos novos desta janela. Contudo e não obstante deste facto, o código poderá ser consultado nos anexos deste projeto.

```
protected void bindImage()
{
    if (tbLivrosRow.ImagemNull())
    {
        btnImg.Content = "Imagen";
        imgBookCover.Source = null;
        return;
    }
    btnImg.Content = "Alterar Imagem";
    string folderPath = "\\\\" + 192.168.1.2 + "\\Biblioteca\\bookImages\\";
    folderPath += tbLivrosRow.Imagem.ToString();
    try
    {
        BitmapImage bitMapImage = new BitmapImage();
        bitMapImage.BeginInit();
        bitMapImage.CacheOption = BitmapCacheOption.None;
        bitMapImage.UriCachePolicy = new
RequestCachePolicy(RequestCacheLevel.BypassCache);
        bitMapImage.CacheOption = BitmapCacheOption.OnLoad;
        bitMapImage.CreateOptions = BitmapCreateOptions.IgnoreImageCache;
        bitMapImage.UriSource = new Uri(folderPath);
        bitMapImage.EndInit();
        imgBookCover.Source = bitMapImage;
    }
    catch (Exception error)
    {
        imgBookCover.Source = null;
    }
}
```

Figura 59: Método *bindImage*

O método *bindImage* tem como função aceder ao caminho de rede da localização da capa do livro e colocar a imagem na aplicação. Este método é chamado durante o método *Bind* e nos eventos associados aos botões de navegação dos registos. Este método verifica inicialmente se a coluna “Imagen” da tabela PDF contém informação, caso não possua o método é terminado, se possuir é criado uma variável do tipo *string* com o caminho de rede para as capas dos livros, e a imagem é carregada numa variável do tipo *BitmapImage*. Sempre que este método é utilizado, e antes de ser carregada uma imagem, a *cache* da aplicação é limpa para garantir que a última versão da capa é a que está a ser visualizada.

```

private void saveDataClick(object sender, RoutedEventArgs e)
{
    if (txtISBN.Text.Length > 0 && cbEditora.SelectedIndex > -1 &&
    txtTitulo.Text.Length > 0)
    {
        string isbn = txtISBN.Text.Replace(" ", String.Empty);
        string editora = cbEditora.SelectedValue.ToString();
        string titulo = txtTitulo.Text.Trim();
        string resumo = txtResumo.Text.Trim();
        if (resumo.Length == 0) resumo = "Resumo não disponível";
        if (value == true)
        {
            try
            {
                tableAdapterLivros.Insert(isbn, titulo,
Convert.ToInt32(editora), null, resumo, null, null);
            }
            catch (Exception error)
            {
                MessageBox.Show(error.Message.ToString(), "Erro",
MessageBoxButton.OK, MessageBoxIcon.Error);
                return;
            }
        }
        else
        {
            try
            {
                tbLivrosRow.ISBN = isbn;
                tbLivrosRow.Titulo = titulo;
                tbLivrosRow.codEditora = Convert.ToInt32(editora);
                tbLivrosRow.Resumo = resumo;
                tableAdapterLivros.Update(dataTableLivros);
            }
            catch (Exception error)
            {
                MessageBox.Show(error.Message.ToString(), "Erro",
MessageBoxButton.OK, MessageBoxIcon.Error);
                return;
            }
        }
        MessageBox.Show("Dados gravados com sucesso!", "Confirmação",
MessageBoxButton.OK, MessageBoxIcon.Information);
        lblMsg.Visibility = Visibility.Hidden;
        btnEliminar.Visibility = Visibility.Visible;
        btnCancelar.Visibility = Visibility.Hidden;
        btnPDF.Visibility = Visibility.Visible;
        btnNewAuthor.IsEnabled = true;
        btnRemoveAuthor.IsEnabled = true;
        btnChangePDF.IsEnabled = true;
        btnImg.IsEnabled = true;
        Bind();
        value = false;
    }
}

```

```

        }
    else
    {
        MessageBox.Show("Não é possível inserir registo nulos. Campos
obrigatórios: ISBN, Editora, Titulo", "Erro", MessageBoxButtons.OK,
MessageBoxImage.Error);
    }
}

```

Figura 60: Event Handler *saveDataClick* (Livros)

O evento *saveDataClick* da janela Livros está associado ao botão “Gravar” e verifica numa fase inicial se os campos de preenchimento obrigatório estão preenchidos, caso não estejam o utilizador é notificado para tal. À imagem de janelas anteriores, este método guarda em variáveis criadas para o efeito os valores dos campos preenchidos, com a particularidade da utilização a propriedade *Trim* e *Replace* para eliminar os espaços em branco que possam existir por lapso do utilizador. Se o valor da variável *value* for verdadeiro, significa que está a ser inserido um novo registo na tabela Livros, assim é inserido os valores na tabela. A atribuição de uma capa, autores e ficheiro PDF apenas pode ser realizado após o livro estar gravado. Caso o valor da variável *value* seja falso, os campos da tabela Livros são atualizados com os novos valores introduzidos pelo utilizador. Convém referir que este método não insere, remove ou actualiza imagem das capas, autores e o ficheiro PDF do livro. Para esse efeito existem eventos próprios que veremos mais à frente.

```

private void deleteClick(object sender, RoutedEventArgs e)
{
    if (lvAutores.Items.Count > 0)
    {
        MessageBox.Show("Remova os autores associados a este livro.", "Erro",
MessageBoxButton.OK, MessageBoxIcon.Error);
        return;
    }
    MessageBoxResult confirm = MessageBox.Show("Tem a certeza que pretende
eliminar este registo?", "Atenção", MessageBoxButtons.YesNo,
MessageBoxImage.Question);
    if (confirm == MessageBoxResult.Yes)
    {
        try
        {
            if (tbLivrosRow.IsImageNull() == false)
            {
                string imageFolderPath =
"\\\\\\192.168.1.2\\Biblioteca\\bookImages\\";

```

```

        string imagePath = imageFolderPath +
tbLivrosRow.Imagem.ToString().Trim();
        if (System.IO.File.Exists(imagePath))
{
    try
    {
        System.IO.File.Delete(imagePath);
    }
    catch (System.IO.IOException error)
    {
        MessageBox.Show("Erro ao eliminar imagem, por favor
contacte o Administrador de Sistema", "Erro", MessageBoxButtons.OK,
MessageBoxImage.Warning);
        return;
    }
}
if (tbLivrosRow.IsPDFNull() == false)
{
    string booksFolderPath =
"\\\\\\192.168.1.2\\Biblioteca\\Books\\";
    string booksFilePath = booksFolderPath +
tbLivrosRow.PDF.ToString().Trim();
    if (System.IO.File.Exists(booksFilePath))
    {
        try
        {
            System.IO.File.Delete(booksFilePath);
        }
        catch (System.IO.IOException error)
        {
            MessageBox.Show("Erro ao eliminar livro, por favor
contacte o Administrador de Sistema", "Erro", MessageBoxButtons.OK,
MessageBoxImage.Warning);
            return;
        }
    }
    tbLivrosRow = (dataSetBiblioteca.LivrosRow)dataTableLivros.Rows[i];
    tbLivrosRow.Delete();
    tableAdapterLivros.Update(dataTableLivros);
}
catch (Exception error)
{
    MessageBox.Show(error.Message.ToString(), "Erro",
MessageBoxButton.OK, MessageBoxIcon.Warning);
    return;
}
MessageBox.Show("Registo eliminado", "Aviso", MessageBoxButtons.OK,
MessageBoxImage.Information);
Bind();
}
}

```

Figura 61: Event Handler *deleteClick* (Livros)

O evento *deleteClick* da janela Livros está associado ao botão eliminar e permite eliminar um livro. O evento quando iniciado verifica se existem autores associados ao livro, caso existam solicita ao utilizador os remova. Seguidamente surge uma caixa de diálogo a solicitar a confirmação da eliminação deste livro. Este evento elimina a imagem da capa e o ficheiro PDF do livro ao contrário do evento associado ao botão “Gravar”, que não lida diretamente com a imagem da capa ou o ficheiro do livro. Este processo elimina permanentemente estes ficheiros não sendo possível recuperar através da reciclagem. No fim surge uma caixa de diálogo com a confirmação da eliminação do livro.

```
private void changeCover(object sender, RoutedEventArgs e)
{
    MessageBoxResult confirm = MessageBox.Show("Esta operação elimina a imagem da
    capa anterior caso exista. Pretende continuar?", "Atenção",
    MessageBoxButton.YesNo, MessageBoxImage.Question);
    if (confirm == MessageBoxResult.No)
        return;
    OpenFileDialog CxDialog = new OpenFileDialog();
    string folderPath = "\\\\" + 192.168.1.2 + "\\Biblioteca\\bookImages\\";
    CxDialog.Title = "Seleccione a capa pretendida";
    CxDialog.Filter = "JPEG (*.jpg)|*.jpg";
    CxDialog.FilterIndex = 1;
    CxDialog.CheckFileExists = true;
    bool? myResult;
    myResult = CxDialog.ShowDialog();
    if (myResult != null && myResult == true)
    {
        if (tbLivrosRow.IsImageNull())
        {
            tbLivrosRow.Imagem = tbLivrosRow.ISBN.ToString() + ".jpg";
            try
            {
                tableAdapterLivros.Update(dataTableLivros);
            }
            catch (Exception error)
            {
                MessageBox.Show(error.Message.ToString(), "Erro",
                MessageBoxButton.OK, MessageBoxImage.Error);
                return;
            }
        }
        string filePath = folderPath + tbLivrosRow.Imagem.ToString().Trim();
        System.IO.File.Copy(CxDialog.FileName, filePath, true);
        imgBookCover.Source = null;
        MessageBox.Show("Imagen inserida com sucesso.", "Aviso",
        MessageBoxButton.OK, MessageBoxImage.Information);
        Bind();
    }
}
```

Figura 62: Event Handler *changeCover*

O evento *changeCover* permite inserir ou alterar a imagem da capa do livro em substituição da capa atual. Para tal o evento alerta o utilizador de que esta operação elimina a capa atual do livro em substituição da nova, caso o utilizador concorde, é aberto uma janela de pesquisa que filtra apenas os ficheiros com a extensão .jpg. Se o livro ainda não possui capa, é inserido o nome da imagem na coluna “Imagen” da tabela Livros. O nome da capa é sempre constituído pelo ISBN, sendo este um número único por livro, permite uma perfeita associação entre a imagem e o livro. Posteriormente a imagem selecionada pelo utilizador é copiada para a localização das imagens substituindo a capa existente. Por último é executado o método *Bind*.

```

private void changePDF(object sender, RoutedEventArgs e)
{
    MessageBoxResult confirm = MessageBox.Show("Esta operação elimina PDF anterior caso exista. Pretende continuar?", "Atenção", MessageBoxButton.YesNo, MessageBoxImage.Question);
    if (confirm == MessageBoxResult.No)
        return;
    OpenFileDialog CxDialog = new OpenFileDialog();
    string folderPath = "\\\\" + 192.168.1.2 + "\\Biblioteca\\Books\\";
    CxDialog.Title = "Seleccione o PDF pretendido";
    CxDialog.Filter = "Adobe PDF Files (*.pdf)|*.pdf";
    CxDialog.FilterIndex = 1;
    CxDialog.CheckFileExists = true;
    bool? myResult;
    myResult = CxDialog.ShowDialog();
    if (myResult != null & myResult == true)
    {
        if (tbLivrosRow.IsPDFNull())
        {
            tbLivrosRow.PDF = tbLivrosRow.ISBN.ToString() + ".pdf";
            try
            {
                tableAdapterLivros.Update(dataTableLivros);
            }
            catch (Exception error)
            {
                MessageBox.Show(error.Message.ToString(), "Erro",
                MessageBoxButton.OK, MessageBoxImage.Error);
                return;
            }
        }
        string filePath = folderPath + tbLivrosRow.PDF.ToString().Trim();
        System.IO.File.Copy(CxDialog.FileName, filePath, true);
        MessageBox.Show("PDF inserido com sucesso.", "Aviso",
        MessageBoxButton.OK, MessageBoxImage.Information);
        Bind();
    }
}

```

Figura 63: Event Handler *changePDF*

O evento *changePDF* representado na figura 63 permite inserir ou atualizar o ficheiro do livro em formato PDF. Este evento está associado ao botão “Alterar PDF”. É possível verificar as similaridades entre o evento *changePDF* e *changeCover*, a diferença reside no tipo de extensão de ficheiro que a janela de pesquisa filtra, sendo neste caso a extensão .pdf.

O botão “Ver PDF” possui um evento associado ao clique com o nome *openPDF*, este evento é idêntico ao descrito no ponto 3.3.6 da janela *searchResults*.

### 3.3.11. Editoras

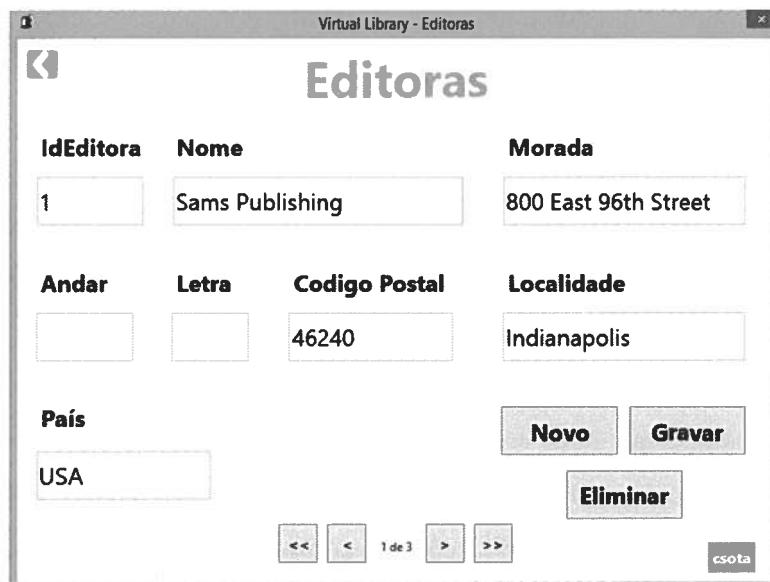


Figura 64: Janela Editoras

A janela Editoras permite adicionar, atualizar e remover uma editora, sendo que estas operações são realizadas sobre a tabela Editoras. É possível verificar a existência mais uma vez dos cinco botões de navegação, já vistos anteriormente em outras janelas da aplicação, bem como os quatro botões que permitem adicionar, remover, gravar e cancelar. Todos os métodos e eventos nesta janela são em grande parte idênticos a outras janelas já revistas. Assim sendo é possível consultar o ponto 3.3.8 para obter uma explicação sobre os métodos e eventos, assim como consultar o código desta janela nos anexos deste projeto.

### 3.3.12. Autores



Figura 65: Janela Autores

A janela Autores é, em contraste com a janela Livros, uma das janelas mais simples desta aplicação. À semelhança das outras janelas de gestão e administração esta possui cinco botões de navegação, bem como os quatro botões que permitem adicionar, remover, gravar e cancelar. O único campo que pode ser alterado nesta janela pelo utilizador é o campo Nome. O campo IdAutor é o campo da chave primária da tabela Autores, e sendo um campo de numeração automática, é a própria aplicação em conjunto com a base de dados que gera este valor. Tal como a janela Editoras no ponto anterior, esta janela possui os métodos e eventos já revistos anteriormente, desta forma é possível consultar o ponto 3.3.8 para obter uma explicação sobre os métodos e eventos, assim como consultar o código desta janela nos anexos deste projeto.

### 3.4. Virtualização

Concluído o desenvolvimento da aplicação, chegou o momento de preparar um laboratório de teste recorrendo às tecnologias de virtualização para testar a aplicação desenvolvida num eventual cenário de implementação em produção.

Desta forma preconizou-se o seguinte cenário para a implementação da aplicação, que tenta recriar da melhor forma a rede de uma organização:

- Servidor controlador de domínio;
- Servidor de SQL;
- Servidor de Hyper-V;
  - Máquina Cliente.

Este laboratório de teste foi realizado num *hypervisor* do tipo2, mais concretamente o *VMware Workstation 11* tendo as máquinas virtuais as seguintes configurações de *hardware*:

Tabela 1: Especificações de *hardware* do laboratório

Máquina Virtual	Processador	Memória	Disco	NIC
DC-Biblioteca	1	2GB	60GB	Host-only
SQL	1	2GB	60GB	Host-only
HyperV	1	4GB	150GB	Host-only

O sistema operativo instalado neste laboratório foi o *Windows Server 2012 R2* da Microsoft, o qual foi instalado nas 3 máquinas descritas na tabela 1. A máquina cliente é uma máquina virtualizada recorrendo à virtualização sobre virtualização ou *nested virtualization* que é possível após a modificação do protocolo OVF (*Open Virtualization Format*) da máquina virtual.

### **3.4.1. Máquina virtual DC-Biblioteca**

Relativamente à máquina virtual DC-Biblioteca, o sistema operativo é o *Windows Server* 2012 R2 como referido sendo este promovido a controlador de domínio. O nome de domínio ficou definido como *library.local* e é neste domínio que as restantes máquinas virtuais farão parte. Após a promoção, foi criado duas contas no *Active Directory Users and Computers*, uma das conta ficou definida como *sqlAdmin* e foi criada numa nova unidade organizacional com o nome *SQL Server Service Accounts* para gerir os serviços inerentes ao SQL Server. A segunda conta é uma conta de utilizador de domínio para simular a autenticação de um utilizador na aplicação.

### **3.4.2. Máquina virtual SQL**

A máquina virtual SQL possui também o sistema operativo *Windows Server* 2012 R2 e foi adicionado ao domínio *library.local*. Sendo esta a máquina responsável pela base de dados da aplicação, foi instalado o *SQL Server* 2012 SP1 e durante a instalação no ecrã de configuração foi definido a conta *sqlAdmin* como a conta com autorização para iniciar os serviços de SQL no servidor. Concluída a instalação foi feito o *attach* da base de dados da aplicação no *SQL Server*.

### **3.4.3. Máquina Virtual HyperV**

A máquina virtual com o nome HyperV, possui a role Hyper-V do *Windows Server* 2012 R2 instalada de forma a virtualizar uma máquina cliente com o sistema operativo *Windows* 8.1.

Para tornar possível a virtualização a *nested virtualization* é necessário efetuar alterações nas configurações da máquina virtual. Em primeiro lugar é necessário editar o ficheiro da máquina virtual com extensão .vmx e introduzir as seguintes alterações:

- *hypervisor.cpuid.v0 = “FALSE”*

- mce.enable = “TRUE”
- vhu.enable = “TRUE”

Em segundo lugar devemos aceder às propriedades (*settings*) da máquina virtual no VMware *Workstation* e proceder às seguintes alterações:

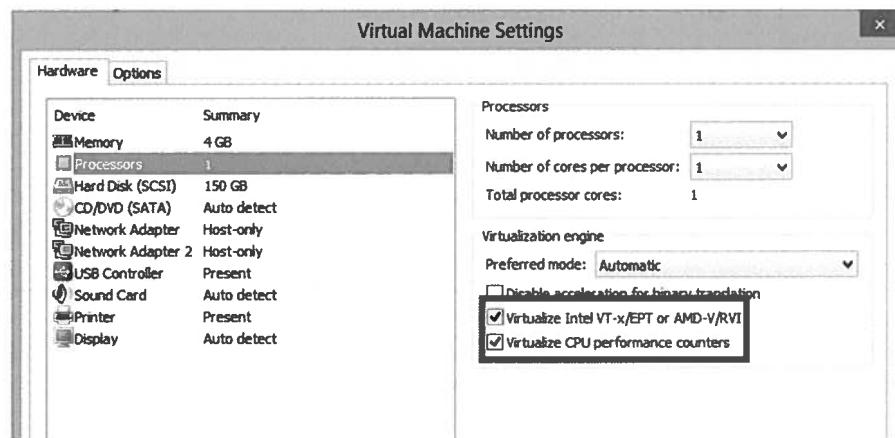


Figura 66: Propriedades da máquina virtual HyperV

### 3.4.4. Máquina virtual cliente

O propósito da máquina virtual cliente é simular um equipamento que funcione como posto de acesso à aplicação numa biblioteca. Utilizando o Hyper-V instalado na máquina virtual HyperV, foi criada uma máquina virtual com sistema operativo Windows 8.1 e que está adicionada ao domínio *library.local*. O quadro seguinte demonstra as características de *hardware* desta máquina.

Tabela 2: Especificações de *hardware* da máquina cliente

Máquina Virtual	Processador	Memória Dinâmica	Disco	NIC
W8	1	512MB - 1024GB	60GB	Host-only

Para permitir a comunicação com a rede do domínio, esta máquina utiliza um *virtual switch* do tipo *external* com o nome *libraryLan*, que está ligado à placa de rede da máquina virtual HyperV.

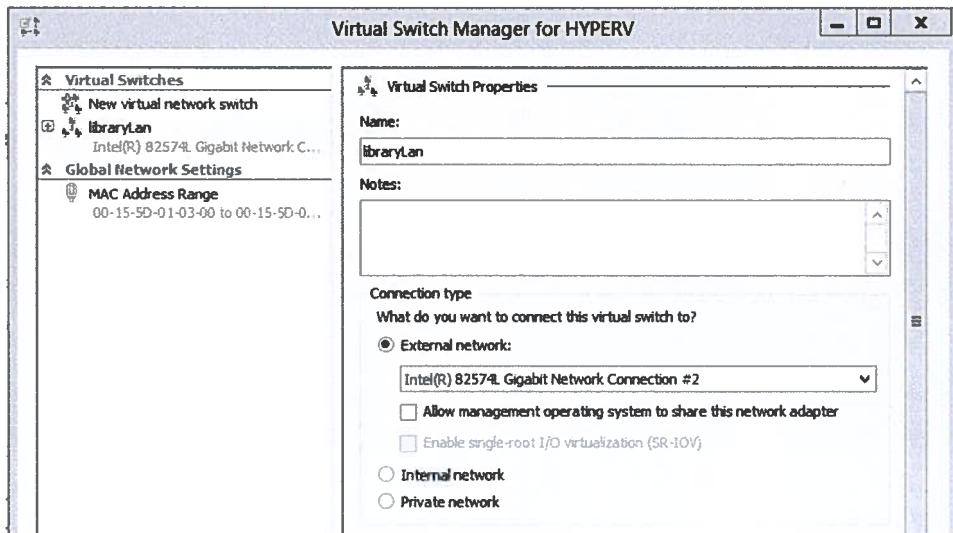


Figura 67: *Virtual Switch Manager* da máquina Hyper-V

O acesso há aplicação é feito através desta máquina cliente que possui o sistema operativo Windows 8.1. Assim o utilizador pode ligar-se a esta máquina com uma ligação de *Remote Desktop Connection* e introduzir as suas credenciais de acesso ao domínio. Uma vez no ambiente de trabalho desta máquina, o utilizador poderá verificar a existência de um ícone de acesso.



Figura 68: Atalho de acesso à aplicação

A partir deste ponto o utilizador deve efetuar um duplo clique para abrir a aplicação e efetuar a autenticação com as credenciais fornecidas pelo administrador de sistema.

## CONCLUSÃO

A realização deste trabalho foi deveras importante no sentido em que me permitiu abordar os conhecimentos adquiridos ao longo da licenciatura e da pós-graduação de uma forma transversal a quase todas as unidades curriculares, utilizando desta forma os conhecimentos adquiridos nomeadamente ao nível das unidades curriculares de Programação, Base de Dados, Sistemas de Gestão de Base de Dados e Administração de Redes.

Tal como exposto na introdução deste projeto, os objetivos passavam pelo desenvolvimento de uma aplicação para uma biblioteca que permitisse a autenticação de utilizadores, pesquisa de livros, consulta de livros e a gestão e administração da aplicação. Outro dos objetivos era utilizar as tecnologias de virtualização para permitir que o acesso à aplicação fosse realizado através de uma máquina virtual garantindo desta forma uma redução nos custos de energia e nos custos dos equipamentos.

Concluído o desenvolvimento da aplicação (capítulo 3) e a criação do laboratório de teste (capítulo 3, ponto 3.4), todos os objetivos propostos foram atingidos. O desenvolvimento da aplicação foi realizado recorrendo às tecnologias .NET, mais em concreto ao ADO.NET, e à linguagem de programação C#, utilizando para esse efeito o Visual Studio 2012 Ultimate. O laboratório de teste foi construído num *hypervisor* do tipo 2, o *VMware Workstation* 11, que permitiu simular a implementação real desta aplicação numa rede organizacional. Contudo o propósito da utilização da virtualização neste projeto era o acesso à aplicação através de uma máquina virtual, para isso foi utilizado a *nested virtualization* que permitiu instalar a *role Hyper-V* no *Windows Server 2012 R2*, o que permite num cenário real a redução de custos energéticos e nos custos de equipamentos na medida em que podem ser utilizado como postos de acesso numa biblioteca máquinas com *hardware* com mais de 10 anos, uma vez que o poder de processamento está centrado num só servidor.

## BIBLIOGRAFIA

*Amazon Elastic Compute Cloud (Amazon EC2).* (1 de 12 de 2014). Obtido de Amazon Web Services:  
<http://aws.amazon.com/pt/ec2/>

Armbrust, M., Fox, A., Griffith, R., Joseph, A. D., Katz, R. H., Konwinski, A., . . . Zaharia, M. (2009).  
*Above the Clouds: A Berkeley View of Cloud.* California: Electrical Engineering and Computer Sciences - University of California.

Beaulieu, A. (2009). *Learning SQL*. California, USA: O'Reilly Media.

Boyce, R. F., Chamberlin, D. D., Hammer, M. M., & King, W. F. (1975). *Specifying queries as relational expressions*. SIGPLAN Not.

Chamberlin, D. D., & Boyce, R. F. (1974). *Sequel: A structured english query language*. FIDET.

Chamberlin, D. D., Astrahan, M. M., Eswaran, K. P., Griffiths, P. P., Lorie, R. A., Mehl, J. W., . . . Wade, B. W. (1976). Sequel 2: A unified approach to data definition, manipulation, and control. *IBM Journal of Research and Development*, 560-575.

Correia, M. P., & Sousa, P. J. (2010). *Segurança no Software*. FCA.

Malik, S. (2005). *Pro ADO.NET 2.0*. USA: Apress.

Microsoft. (2013). Microsoft Azure for Research Overview. *Microsoft Azure for Research Overview*, 1-8.

O'Reilly, T. (4 de Maio de 2014). *Cloud Century*. Obtido de Web site de Cloud Century:  
<http://cloudcentury.com/index.html>

Patrick, T. (2010). *Microsoft ADO.NET 4 Step by Step*. California: O'Reilly Media, Inc.

Ramakrishnan, R., & Gehrke, J. (s.d.). *Database Management Systems*, 3rd Edition.

Vietstack. (10 de 12 de 2014). *Introduction of Virtualization*. Obtido de Vietstack:  
<http://vietstack.wordpress.com/2014/10/03/introduction-of-virtualization/>

W3Schools. (23 de 11 de 2014). *Refsnes Data*. Obtido de W3Schools:  
<http://www.w3schools.com/sql/default.asp>

Wang, X., Wang, B., & Haung, J. (2011). Wang. *2011 IEEE International Conference on Computer Science and Automation Engineering (CSAE)* (pp. 404 - 410). Shanghai: IEEE.

# **ANEXOS**

## Anexo I - Classe Cripto.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Security.Cryptography;
using System.Text;
using System.Threading.Tasks;

namespace Biblioteca
{
    public static class Cripto
    {
        public static string sha256(string password) // recebe e cria um hash da
palavra-passe
        {
            SHA256Managed crypt = new SHA256Managed();
            string hash = String.Empty;
            byte[] crypto = crypt.ComputeHash(Encoding.ASCII.GetBytes(password), 0,
Encoding.ASCII.GetByteCount(password));
            foreach (byte bit in crypto)
            {
                hash += bit.ToString("x2");
            }
            return hash;
        }
    }
}
```

## Anexo II - Janela Main Window

```

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.Data;
using System.Data.SqlClient;
using System.IO;
using Biblioteca.DataBase.dataSetBibliotecaTableAdapters;
using Biblioteca.DataBase;

namespace Biblioteca
{
    public partial class MainWindow : Window
    {
        UtilizadoresTableAdapter tableAdapterUtilizadores = new
UtilizadoresTableAdapter();
        dataSetBiblioteca.UtilizadoresDataTable dataTableUtilizadores;
        dataSetBiblioteca.UtilizadoresRow tbUtilizadoresRow;

        public MainWindow()
        {
            InitializeComponent();
            txtUser.Focus();
        }

        private void btnLogin_Click_1(object sender, RoutedEventArgs e) // efetua o
login na aplicação
        {
            if (txtUser.Text.Length > 0)
            {
                dataTableUtilizadores =
tableAdapterUtilizadores.getDataByUserName(txtUser.Text.ToString());
                tbUtilizadoresRow =
(dataSetBiblioteca.UtilizadoresRow)dataTableUtilizadores[0];
                string typedPassword = Cripto.sha256(txtPassword.Password);
                string storedPassword = tbUtilizadoresRow.userPassword;
                if (string.Compare(typedPassword, storedPassword) == 0)
                {
                    dataTableUtilizadores.Clear();
                    string userName = txtUser.Text;
                    Home homepage = new Home(userName);
                    homepage.Show();
                    this.Close();
                }
                else
                {
                    lblMsg.Visibility = Visibility.Visible;
                    lblMsg.Content = "Utilizador ou palavra-passe incorrecta.";
                }
            }
            else
            {

```

```
        lblMsg.Visibility = Visibility.Visible;
        lblMsg.Content = "Preencha os campos Username e Password.";
    }
}

private void btnPassLost_Click_1(object sender, RoutedEventArgs e) // acede à
janela de recuperação de palavra-passe
{
    if (txtUser.Text.Length > 0)
    {
        dataTableUtilizadores =
tableAdapterUtilizadores.getDataByUserName(txtUser.Text.ToString());
        if (dataTableUtilizadores.Rows.Count > 0)
        {
            string userName = txtUser.Text;
            PasswordRecover passRecover = new PasswordRecover(userName);
            txtPassword.Clear();
            passRecover.ShowDialog();
        }
        else
        {
            lblMsg.Visibility = Visibility.Visible;
            lblMsg.Content = "Introduza um utiliador válido.";
        }
    }
    else
    {
        lblMsg.Visibility = Visibility.Visible;
        lblMsg.Content = "Preencha o campo Username.";
    }
}
}
```

## Anexo III - Janela PasswordRecover

```

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;
using System.Data;
using System.Data.SqlClient;
using System.IO;
using Biblioteca.DataBase.dataSetBibliotecaTableAdapters;
using Biblioteca.DataBase;

namespace Biblioteca
{
    public partial class PasswordRecover : Window
    {
        UtilizadoresTableAdapter tableAdapterUtilizadores = new
UtilizadoresTableAdapter();
        dataSetBiblioteca.UtilizadoresDataTable dataTableUtilizadores;
        dataSetBiblioteca.UtilizadoresRow tbUtilizadoresRow;
        string userName;

        public PasswordRecover(string _userName)
        {
            InitializeComponent();
            txtPassword.Focus();
            lblUser.Content = userName = _userName;
        }

        private void btnClean_Click_1(object sender, RoutedEventArgs e) // limpa as
caixas de texto
        {
            txtPassword.Clear();
            txtNewPassword.Clear();
            txtNewPassword1.Clear();
        }

        private void btnConfirm_Click_1(object sender, RoutedEventArgs e) // altera a
palavra-passe
        {
            if (txtPassword.Password.Length > 0 && txtNewPassword.Password.Length > 0
&& txtNewPassword1.Password.Length > 0)
            {
                if (txtNewPassword.Password == txtNewPassword1.Password)
                {
                    dataTableUtilizadores =
tableAdapterUtilizadores.getDataByUserName(lblUser.Content.ToString());
                    tbUtilizadoresRow =
(dataSetBiblioteca.UtilizadoresRow)dataTableUtilizadores.Rows[0];
                    string typedPassword = Cripto.sha256(txtPassword.Password);
                    string storedPassword = tbUtilizadoresRow.userPassword;
                    if (string.Compare(typedPassword, storedPassword) == 0)
                    {
                        string password = Cripto.sha256(txtNewPassword.Password);
                        tbUtilizadoresRow.userPassword = password;
                    }
                }
            }
        }
    }
}

```

```
        tableAdapterUtilizadores.Update(dataTableUtilizadores);
        dataTableUtilizadores.AcceptChanges();
        MessageBoxResult msgSuccess = MessageBox.Show("Palavra-passe
alterada com sucesso!", "Confirmação");
        this.Close();
    }
    else
    {
        lblMsg.Visibility = Visibility.Visible;
        lblMsg.Content = "Palavra-passe incorrecta.";
    }
}
else
{
    lblMsg.Visibility = Visibility.Visible;
    lblMsg.Content = "Os dados introduzidos estão incorrectos.";
}
else
{
    lblMsg.Visibility = Visibility.Visible;
    lblMsg.Content = "É necessário preencher todos os campos.";
}
}
}
}
```

## Anexo IV - Janela Home

```

using Biblioteca.DataBase;
using Biblioteca.DataBase.dataSetBibliotecaTableAdapters;
using System;
using System.Collections.Generic;
using System.Data.SqlClient;
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 Biblioteca
{
    public partial class Home : Window
    {
        UtilizadoresTableAdapter tableAdapterUtilizadores = new
        UtilizadoresTableAdapter();
        dataSetBiblioteca.UtilizadoresDataTable dataTableUtilizadores;
        dataSetBiblioteca.UtilizadoresRow tbUtilizadoresRow;

        LivrosTableAdapter tableAdapterLivros = new LivrosTableAdapter();
        dataSetBiblioteca.LivrosDataTable dataTableLivros;
        dataSetBiblioteca.LivrosRow tbLivrosRow;

        private SqlConnection sqlConn = new SqlConnection();
        private System.Data.DataSet dataSet = new System.Data.DataSet();
        private System.Data.DataTable dataTable;

        string userName;
        int searchOption;

        public Home(string _userName)
        {
            InitializeComponent();
            userName = _userName;
            lblUser.Content = userName;
            txtSearch.Focus();
            dataTableUtilizadores =
            tableAdapterUtilizadores.getDataByUserName(userName);
            tbUtilizadoresRow =
            (dataSetBiblioteca.UtilizadoresRow)dataTableUtilizadores.Rows[0];
            if (tbUtilizadoresRow.tipoUtilizador.Equals("C"))
            {
                dockPanel.Visibility = Visibility.Hidden;
            }
        }

        private SqlCommand search(string searchParam, int searchOption) // cria um
array com os parametros de pesquisa
        {
            SqlCommand command = new SqlCommand();
            string sql;
            string[] allWords = searchParam.Split(new char[] { ' ' },
StringSplitOptions.RemoveEmptyEntries);
            if (searchOption == 1)

```

```

        {
            sql = "SELECT Livros.ISBN, Livros.Titulo, Livros.Tema, Livros.Resumo,
Livros.Imagem, Livros.fotoTipo, Livros.PDF, Editoras.Nome AS nomeEditora FROM Livros
INNER JOIN Editoras ON Livros.codEditora = Editoras.codEditora WHERE ";
        }
        else
        {
            sql = "SELECT Livros.ISBN, Livros.Titulo, Livros.Tema, Livros.Resumo,
Livros.Imagem, Livros.fotoTipo, Livros.PDF, Editoras.Nome AS nomeEditora FROM Livros
INNER JOIN livrosAutores ON Livros.ISBN = livrosAutores.ISBN INNER JOIN Autores ON
livrosAutores.idAutor = Autores.idAutor INNER JOIN Editoras ON Livros.codEditora =
Editoras.codEditora WHERE ";
        }
        using (command)
        {
            for (int i = 0; i < allWords.Length; ++i)
            {
                if (i > 0)
                {
                    sql += "OR ";
                }

                if (searchOption == 1)
                {
                    sql += string.Format("(Livros.Titulo LIKE '%{0}%') ", allWords[i]);
                }
                else
                {
                    sql += string.Format("(Autores.Nome LIKE '%{0}%') ", allWords[i]);
                }
            }
            command.CommandText = sql;
        }
        return command;
    }

    private void btnSearch_MouseLeftButtonUp_1(object sender, MouseButtonEventArgs e) // efetua uma pesquisa
    {
        if (txtSearch.Text.Trim().Length > 0)
        {
            if(txtSearch.Text.Trim().Length > 3)
            {
                if (rbSearchOption1.IsChecked.Value)
                {
                    searchOption = 1;
                    sqlConn.ConnectionString =
Properties.Settings.Default.BibliotecaConnectionString;
                    string connectionString = sqlConn.ConnectionString.ToString();
                    SqlDataAdapter sqlDataAdapter = new
SqlDataAdapter(search(txtSearch.Text, searchOption).CommandText, connectionString);
                    sqlDataAdapter.Fill(dataSet, "livrosTitulo");
                    dataTable = dataSet.Tables["livrosTitulo"];
                }
                else
                {
                    searchOption = 2;
                    sqlConn.ConnectionString =
Properties.Settings.Default.BibliotecaConnectionString;
                    string connectionString = sqlConn.ConnectionString.ToString();

```

```

        SqlDataAdapter sqlDataAdapter = new
SqlDataAdapter(search(txtSearch.Text, searchOption).CommandText, connectionString);
        sqlDataAdapter.Fill(dataSet, "livrosTitulo");
        dataTable = dataSet.Tables["livrosTitulo"];
    }
    if (dataTable.Rows.Count > 0)
    {
        searchResults searchResults = new searchResults(userName,
search(txtSearch.Text, searchOption));
        searchResults.Show();
        this.Close();
    }
    else
    {
        MessageBox.Show("A pesquisa por '" + txtSearch.Text + "' não
retornou quaisquer resultados","Aviso");
    }
}
else
{
    lblMsg.Content = "Elemento de pesquisa introduzido insuficiente.";
    lblMsg.Visibility = Visibility.Visible;
}
}
else
{
    lblMsg.Content = "É necessário preencher o campo de pesquisa.";
    lblMsg.Visibility = Visibility.Visible;
}
}

private void menuItemTiposPessoa_Click_1(object sender, RoutedEventArgs e) // abre a janela TiposPessoa
{
    tipoPessoa tipoPessoa = new tipoPessoa(userName);
    tipoPessoa.Show();
    this.Close();
}

private void menuItemUtilizadores_Click_1(object sender, RoutedEventArgs e) // abre a janela Utilizadores
{
    Utilizadores users = new Utilizadores(userName);
    users.Show();
    this.Close();
}

private void menuItemPessoas_Click_1(object sender, RoutedEventArgs e) // abre a janela Pessoas
{
    Pessoas pessoas = new Pessoas(userName);
    pessoas.Show();
    this.Close();
}

private void menuItemEditoras_Click_1(object sender, RoutedEventArgs e) // abre a janela Editoras
{
    Editoras editoras = new Editoras(userName);
    editoras.Show();
    this.Close();
}

```

```
    private void menuItemLivros_Click_1(object sender, RoutedEventArgs e) // abre
a janela Livros
{
    Livros livros = new Livros(userName);
    livros.Show();
    this.Close();
}

    private void menuItemAutores_Click_1(object sender, RoutedEventArgs e) // abre
a janela Autores
{
    Autores autores = new Autores(userName);
    autores.Show();
    this.Close();
}

    private void lblAdvancedSearch_PreviewMouseLeftButtonDown_1(object sender,
MouseButtonEventArgs e) // abre a janela de pesquisa avançada
{
    advancedSearch advancedSearch = new advancedSearch(userName);
    advancedSearch.Show();
    this.Close();
}
}
```

## Anexo V - Janela AdvancedSearch

```

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;
using System.Data;
using Biblioteca DataBase;
using Biblioteca DataBase.dataSetBibliotecaTableAdapters;
using System.Data.SqlClient;

namespace Biblioteca
{
    public partial class advancedSearch : Window
    {

        private SqlConnection sqlConn = new SqlConnection();
        private System.Data.DataSet dataSet = new System.Data.DataSet();
        private System.Data.DataTable dataTable;

        string userName;

        public advancedSearch(string _userName)
        {
            InitializeComponent();
            userName = _userName;
            lblUser.Content = userName;
        }

        private string[] searchArray(string searchParam) // remove os espaços entre os
parametros de pesquisa
        {
            string[] allWords = searchParam.Split(new char[] { ' ' },
StringSplitOptions.RemoveEmptyEntries);
            return allWords;
        }

        private SqlCommand search() // cria um array com os parametros de pesquisa
        {
            SqlCommand command = new SqlCommand();
            string sql = "SELECT DISTINCT Livros.ISBN, Livros.Titulo, Livros.Tema,
Livros.Resumo, Livros.Imagem, Livros.fotoTipo, Livros.PDF, Editoras.Nome AS
nomeEditora FROM Livros INNER JOIN livrosAutores ON Livros.ISBN = livrosAutores.ISBN
INNER JOIN Autores ON livrosAutores.idAutor = Autores.idAutor INNER JOIN Editoras ON
Livros.codEditora = Editoras.codEditora WHERE (";
            string[] titulo = searchArray(txtTitulo.Text);
            string[] autor = searchArray(txtAutor.Text);
            string[] editora = searchArray(txtEditora.Text);
            using (command)
            {
                if (txtTitulo.Text.Trim().Length > 0)
                {
                    for (int i = 0; i < titulo.Length; ++i)
                    {
                        if (i > 0)

```

```

        {
            sql += "OR ";
        }
        sql += string.Format("(Livros.Titulo LIKE '%{0}%') ",
    titulo[i]);
    }
}
if (txtAutor.Text.Trim().Length > 0)
{
    if (cbOption1.SelectedIndex == 0 && txtTitulo.Text.Trim().Length >
0)
        sql += ") AND (";
    if (cbOption1.SelectedIndex == 1 && txtTitulo.Text.Trim().Length >
0)
        sql += ") OR (";
    for (int i = 0; i < autor.Length; ++i)
    {
        if (i > 0)
        {
            sql += "OR ";
        }
        sql += string.Format("(Autores.Nome LIKE '%{0}%') ",
    autor[i]);
    }
}
if (txtEditora.Text.Trim().Length > 0)
{
    if (cbOption2.SelectedIndex == 0 && (txtAutor.Text.Trim().Length >
0 || txtTitulo.Text.Trim().Length > 0))
        sql += ") AND (";
    if (cbOption2.SelectedIndex == 1 && (txtAutor.Text.Trim().Length >
0 || txtTitulo.Text.Trim().Length > 0))
        sql += ") OR (";
    for (int i = 0; i < editora.Length; ++i)
    {
        if (i > 0)
        {
            sql += "OR ";
        }
        sql += string.Format("(Editoras.Nome LIKE '%{0}%') ",
    editora[i]);
    }
}
sql += ")";
command.CommandText = sql;
}
return command;
}

private void lblSimpleSearch_PreviewMouseLeftButtonDown_1(object sender,
MouseEventArgs e) // regressa à janela Home
{
    Home home = new Home(userName);
    home.Show();
    this.Close();
}

private void btnSearch_MouseLeftButtonUp_1(object sender, MouseEventArgs
e) // efetua uma pesquisa
{

```

```

        if (txtTitulo.Text.Trim().Length > 0 || txtEditora.Text.Trim().Length > 0
        || txtAutor.Text.Trim().Length > 0)
        {
            try
            {
                sqlConn.ConnectionString =
Properties.Settings.Default.BibliotecaConnectionString;
                string connectionString = sqlConn.ConnectionString.ToString();
                SqlDataAdapter sqlDataAdapter = new
SqlDataAdapter(search().CommandText, connectionString);
                sqlDataAdapter.Fill(dataSet, "livrosTitulo");
                dataTable = dataSet.Tables["livrosTitulo"];
            }
            catch (Exception error)
            {
                MessageBox.Show(error.Message.ToString(), "Erro",
MessageBoxButton.OK, MessageBoxIcon.Error);
                return;
            }
            if (dataTable.Rows.Count > 0)
            {
                searchResults searchResults = new searchResults(userName,
search());
                searchResults.Show();
                this.Close();
            }
            else
            {
                MessageBox.Show("A pesquisa não retornou quaisquer resultados",
"Aviso", MessageBoxButton.OK, MessageBoxIcon.Information);
            }
        }
        else
        {
            lblMsg.Content = "Elemento de pesquisa introduzido insuficiente.";
            lblMsg.Visibility = Visibility.Visible;
        }
    }
}

```

## Anexo VI - Janela searchResults

```

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;
using Biblioteca.DataBase;
using Biblioteca.DataBase.dataSetBibliotecaTableAdapters;
using System.Data.SqlClient;
using System.Configuration;

namespace Biblioteca
{
    public partial class searchResults : Window
    {
        LivrosTableAdapter tableAdapterLivros = new LivrosTableAdapter();
        dataSetBiblioteca.LivrosDataTable dataTableLivros;
        dataSetBiblioteca.LivrosRow tbLivrosRow;

        AutoresTableAdapter tableAdapterAutores = new AutoresTableAdapter();
        dataSetBiblioteca.AutoresDataTable dataTableAutores;

        private SqlConnection sqlConn = new SqlConnection();
        private System.Data.DataSet dataSet = new System.Data.DataSet();
        private System.Data.DataTable dataTable;
        private System.Data.DataRow dataRow;

        string userName;
        SqlCommand sqlCommand = new SqlCommand();
        int previousClickedRow = -1;

        Style rowStyle = new Style(typeof(DataGridRow));

        public searchResults(string _userName, SqlCommand _command)
        {
            InitializeComponent();
            userName = _userName;
            lblUser.Content = userName;
            sqlCommand = _command;
            dataGrid.CanUserAddRows = false;
            Bind();
            rowStyle.Setters.Add(new EventSetter(DataGridRow.MouseDoubleClickEvent,
new MouseButtonEventHandler(rowDoubleClick)));
            dataGrid.RowStyle = rowStyle;
        }

        private void Image_MouseLeftButtonDown_1(object sender, MouseButtonEventArgs e) // regressa à janela Home
        {
            Home home = new Home(userName);
            this.Close();
            home.Show();
        }
    }
}

```

```

    private void HandleCanExecute(object sender, CanExecuteRoutedEventArgs e) // impede a cópia não autorizada do resumo do livro
    {
        if (e.Command == ApplicationCommands.Cut || e.Command == ApplicationCommands.Copy || e.Command == ApplicationCommands.Paste)
        {
            e.CanExecute = false;
            e.Handled = true;
        }
    }

    protected void Bind() // preenche a lista de pesquisa com os resultados
    {
        sqlConn.ConnectionString =
Properties.Settings.Default.BibliotecaConnectionString;
        string connectionString = sqlConn.ConnectionString.ToString();
        SqlDataAdapter sqlDataAdapter = new SqlDataAdapter(sqlCommand.CommandText,
connectionString);
        sqlDataAdapter.Fill(dataSet, "livrosTitulo");
        dataTable = dataSet.Tables["livrosTitulo"];
        dataGrid.DataContext = dataTable.DefaultView;
    }

    private void rowDoubleClick(object sender, MouseButtonEventArgs e) // permite um duplo clique nos resultados e visualizar a informação do livro
    {
        DataGridRow row = sender as DataGridRow;
        if (row.GetIndex() != previousClickedRow)
        {
            DataRow dataRow = dataTable.Rows[row.GetIndex()];
            if (dataRow.ItemArray[4] != DBNull.Value)
            {
                string bookCover = "\\\\" + 192.168.1.2 + "\\Biblioteca\\bookImages\\" +
dataRow.ItemArray[4].ToString();
                try
                {
                    BitmapImage bitMapImage = new BitmapImage();
                    bitMapImage.BeginInit();
                    bitMapImage.UriSource = new Uri(bookCover);
                    bitMapImage.CacheOption = BitmapCacheOption.OnLoad;
                    imgBookCover.Source = bitMapImage;
                    bitMapImage.EndInit();
                }
                catch (Exception error)
                {
                    imgBookCover.Source = null;
                }
            }
            dataTableAutores =
tableAdapterAutores.getDataByAutorISBN(dataRow.ItemArray[0].ToString());
            txtClick.Visibility = Visibility.Hidden;
            border.Visibility = Visibility.Hidden;
            stackAutor.Children.Clear();
            foreach (DataSetBiblioteca.AutoresRow tbAutoresRow in
dataTableAutores.Rows)
            {
                TextBlock text = new TextBlock();
                text.Text = tbAutoresRow.Nome;
                text.FontSize = 20;
                stackAutor.Children.Add(text);
            }
            if (dataTableAutores.Rows.Count == 1)

```



## Anexo VII - Janela Pessoas

```

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;
using System.Data;
using Biblioteca.DataBase;
using Biblioteca.DataBase.dataSetBibliotecaTableAdapters;
using System.Data.SqlClient;
using System.Configuration;

namespace Biblioteca
{
    public partial class Pessoas : Window
    {
        PessoasTableAdapter tableAdapterPessoas = new PessoasTableAdapter();
        dataSetBiblioteca.PessoasDataTable dataTablePessoas;
        dataSetBiblioteca.PessoasRow tbPessoasRow;

        TiposPessoaTableAdapter tableAdapterTiposPessoa = new
        TiposPessoaTableAdapter();
        dataSetBiblioteca.TiposPessoaDataTable dataTableTiposPessoa;

        EmailsTableAdapter tableAdapterEmails = new EmailsTableAdapter();
        dataSetBiblioteca.EmailsDataTable dataTableEmails;

        TelefonesTableAdapter tableAdapterTelefone = new TelefonesTableAdapter();
        dataSetBiblioteca.TelefonesDataTable dataTableTelefone;

        string userName;
        bool value = false;
        int i = 0;

        public Pessoas(string _userName)
        {
            InitializeComponent();
            userName = _userName;
            lblUser.Content = userName;
            Bind();
        }

        protected void Bind() // preenche os campos da janela
        {
            i = 0;
            dataTablePessoas = tableAdapterPessoas.getPessoas();
            tbPessoasRow = (dataSetBiblioteca.PessoasRow)dataTablePessoas.Rows[0];
            lblReg.Content = (i + 1).ToString() + " de " +
            dataTablePessoas.Count().ToString();
            txtIdPessoa.Text = tbPessoasRow.idPessoa.ToString();
            fillcbPessoa();
            fillListView();
            cbPessoa.SelectedValue = tbPessoasRow.Tipo.ToString();
            if(tbPessoasRow.IsNomeNull() == false)
                txtNome.Text = tbPessoasRow.Nome.ToString();
        }
    }
}

```

```

        if(tbPessoasRow.IsCCNull() == false)
            txtCC.Text = tbPessoasRow.CC.ToString();
        if(tbPessoasRow.IsRuaNull() == false)
            txtRua.Text = tbPessoasRow.Rua.ToString();
        if(tbPessoasRow.IsAndarNull() == false)
            txtAndar.Text = tbPessoasRow.Andar.ToString();
        if (tbPessoasRow.IsLetraNull() == false)
            txtLetra.Text = tbPessoasRow.Letra.ToString();
        if(tbPessoasRow.IscodPostalNull() == false)
            txtCodPostal.Text = tbPessoasRow.codPostal.ToString();
        if(tbPessoasRow.IsLocalidadeNull() == false)
            txtLocalidade.Text = tbPessoasRow.Localidade.ToString();
        if(tbPessoasRow.IsPaisNull() == false)
            txtPais.Text = tbPessoasRow.Pais.ToString();
    }

    protected void clearValues() // limpa os valores dos campos
    {
        txtNome.Clear();
        txtCC.Clear();
        txtRua.Clear();
        txtAndar.Clear();
        txtLetra.Clear();
        txtCodPostal.Clear();
        txtLocalidade.Clear();
        txtPais.Clear();
    }

    protected void fillcbPessoa() // preenche a lista de valores TiposPessoa
    {
        dataTableTiposPessoa = tableAdapterTiposPessoa.getTipoPessoa();
        cbPessoa.ItemsSource = dataTableTiposPessoa;
        cbPessoa.DisplayMemberPath = "Desig";
        cbPessoa.SelectedValuePath = "idTipo";
    }

    protected void fillListView() // preenche a lista de valores com os emails e
telefones
    {
        dataTableEmails =
tableAdapterEmails.getDataByIdPessoa(tbPessoasRow.idPessoa);
        dataTableTelefone =
tableAdapterTelefone.getDataByIdPessoa(tbPessoasRow.idPessoa);
        lvEmail.DataContext = dataTableEmails.DefaultView;
        lvContact.DataContext = dataTableTelefone.DefaultView;
    }

    private void Image_MouseLeftButtonDown_1(object sender, MouseButtonEventArgs e) // regressa à janela Home
    {
        Home home = new Home(userName);
        this.Close();
        home.Show();
    }

    private void btnFirst_Click_1(object sender, RoutedEventArgs e) // avança para
o primeiro registo
    {
        if (value == true)
            return;
        i = 0;
        tbPessoasRow = (dataSetBiblioteca.PessoasRow)dataTablePessoas.Rows[0];
    }
}

```

```

lblReg.Content = "1 de " + dataTablePessoas.Count().ToString();
txtIdPessoa.Text = tbPessoasRow.idPessoa.ToString();
cbPessoa.SelectedValue = tbPessoasRow.Tipo.ToString();
fillListView();
clearValues();
if (tbPessoasRow.IsNomeNull() == false)
    txtNome.Text = tbPessoasRow.Nome.ToString();
if (tbPessoasRow.IsCCNull() == false)
    txtCC.Text = tbPessoasRow.CC.ToString();
if (tbPessoasRow.IsRuaNull() == false)
    txtRua.Text = tbPessoasRow.Rua.ToString();
if (tbPessoasRow.IsAndarNull() == false)
    txtAndar.Text = tbPessoasRow.Andar.ToString();
if (tbPessoasRow.IsLetraNull() == false)
    txtLetra.Text = tbPessoasRow.Letra.ToString();
if (tbPessoasRow.IscodPostalNull() == false)
    txtCodPostal.Text = tbPessoasRow.codPostal.ToString();
if (tbPessoasRow.IsLocalidadeNull() == false)
    txtLocalidade.Text = tbPessoasRow.Localidade.ToString();
if (tbPessoasRow.IsPaisNull() == false)
    txtPais.Text = tbPessoasRow.Pais.ToString();
}

private void btnPrevious_Click_1(object sender, RoutedEventArgs e) // avança
para o registo anterior
{
    if (i <= 0 || value == true)
        return;
    i--;
    tbPessoasRow = (dataSetBiblioteca.PessoasRow)dataTablePessoas.Rows[i];
    lblReg.Content = (i + 1).ToString() + " de " +
    dataTablePessoas.Count().ToString();
    txtIdPessoa.Text = tbPessoasRow.idPessoa.ToString();
    cbPessoa.SelectedValue = tbPessoasRow.Tipo.ToString();
    fillListView();
    clearValues();
    if (tbPessoasRow.IsNomeNull() == false)
        txtNome.Text = tbPessoasRow.Nome.ToString();
    if (tbPessoasRow.IsCCNull() == false)
        txtCC.Text = tbPessoasRow.CC.ToString();
    if (tbPessoasRow.IsRuaNull() == false)
        txtRua.Text = tbPessoasRow.Rua.ToString();
    if (tbPessoasRow.IsAndarNull() == false)
        txtAndar.Text = tbPessoasRow.Andar.ToString();
    if (tbPessoasRow.IsLetraNull() == false)
        txtLetra.Text = tbPessoasRow.Letra.ToString();
    if (tbPessoasRow.IscodPostalNull() == false)
        txtCodPostal.Text = tbPessoasRow.codPostal.ToString();
    if (tbPessoasRow.IsLocalidadeNull() == false)
        txtLocalidade.Text = tbPessoasRow.Localidade.ToString();
    if (tbPessoasRow.IsPaisNull() == false)
        txtPais.Text = tbPessoasRow.Pais.ToString();
}

private void btnNext_Click_1(object sender, RoutedEventArgs e) // avança para
o registo seguinte
{
    if (i >= dataTablePessoas.Count() - 1 || value == true)
        return;
    i++;
    tbPessoasRow = (dataSetBiblioteca.PessoasRow)dataTablePessoas.Rows[i];
}

```

```

        lblReg.Content = (i + 1).ToString() + " de " +
dataTablePessoas.Count().ToString();
        txtIdPessoa.Text = tbPessoasRow.idPessoa.ToString();
        cbPessoa.SelectedValue = tbPessoasRow.Tipo.ToString();
        fillListView();
        clearValues();
        if (tbPessoasRow.IsNomeNull() == false)
            txtNome.Text = tbPessoasRow.Nome.ToString();
        if (tbPessoasRow.IsCCNull() == false)
            txtCC.Text = tbPessoasRow.CC.ToString();
        if (tbPessoasRow.IsRuaNull() == false)
            txtRua.Text = tbPessoasRow.Rua.ToString();
        if (tbPessoasRow.IsAndarNull() == false)
            txtAndar.Text = tbPessoasRow.Andar.ToString();
        if (tbPessoasRow.IsLetraNull() == false)
            txtLetra.Text = tbPessoasRow.Letra.ToString();
        if (tbPessoasRow.IscodPostalNull() == false)
            txtCodPostal.Text = tbPessoasRow.codPostal.ToString();
        if (tbPessoasRow.IsLocalidadeNull() == false)
            txtLocalidade.Text = tbPessoasRow.Localidade.ToString();
        if (tbPessoasRow.IsPaisNull() == false)
            txtPais.Text = tbPessoasRow.Pais.ToString();
    }

    private void btnLast_Click_1(object sender, RoutedEventArgs e) // avança para
o último registo
    {
        if (value == true)
            return;
        i = dataTablePessoas.Count() - 1;
        tbPessoasRow =
(dataSetBiblioteca.PessoasRow)dataTablePessoas.Rows[dataTablePessoas.Count() - 1];
        lblReg.Content = dataTablePessoas.Count().ToString() + " de " +
dataTablePessoas.Count().ToString();
        txtIdPessoa.Text = tbPessoasRow.idPessoa.ToString();
        cbPessoa.SelectedValue = tbPessoasRow.Tipo.ToString();
        fillListView();
        clearValues();
        if (tbPessoasRow.IsNomeNull() == false)
            txtNome.Text = tbPessoasRow.Nome.ToString();
        if (tbPessoasRow.IsCCNull() == false)
            txtCC.Text = tbPessoasRow.CC.ToString();
        if (tbPessoasRow.IsRuaNull() == false)
            txtRua.Text = tbPessoasRow.Rua.ToString();
        if (tbPessoasRow.IsAndarNull() == false)
            txtAndar.Text = tbPessoasRow.Andar.ToString();
        if (tbPessoasRow.IsLetraNull() == false)
            txtLetra.Text = tbPessoasRow.Letra.ToString();
        if (tbPessoasRow.IscodPostalNull() == false)
            txtCodPostal.Text = tbPessoasRow.codPostal.ToString();
        if (tbPessoasRow.IsLocalidadeNull() == false)
            txtLocalidade.Text = tbPessoasRow.Localidade.ToString();
        if (tbPessoasRow.IsPaisNull() == false)
            txtPais.Text = tbPessoasRow.Pais.ToString();
    }

    private void insertDataClick(object sender, RoutedEventArgs e) // prepara o
ambiente para a inserção de um novo registo
    {
        lblMsg.Visibility = Visibility.Visible;
        btnEliminar.Visibility = Visibility.Hidden;
        btnCancela.Visibility = Visibility.Visible;

```

```

        lblEmail.Visibility = Visibility.Hidden;
        lblContact.Visibility = Visibility.Hidden;
        txtEmail.Visibility = Visibility.Hidden;
        txtContact.Visibility = Visibility.Hidden;
        clearValues();
        cbPessoa.SelectedIndex = -1;
        txtIdPessoa.Clear();
        lvEmail.DataContext = null;
        lvContact.DataContext = null;
        value = true;
    }

    private void cancelClick(object sender, RoutedEventArgs e) // cancela a
introdução de um novo registo
    {
        lblMsg.Visibility = Visibility.Hidden;
        btnEliminar.Visibility = Visibility.Visible;
        btnCancelar.Visibility = Visibility.Hidden;
        lblEmail.Visibility = Visibility.Visible;
        lblContact.Visibility = Visibility.Visible;
        txtEmail.Visibility = Visibility.Visible;
        txtContact.Visibility = Visibility.Visible;
        Bind();
        value = false;
    }

    private void saveDataClick(object sender, RoutedEventArgs e) // guarda as
alterações e os novos registo na base de dados
    {
        if (cbPessoa.SelectedIndex > -1 && txtNome.Text.Length > 0 &&
txtCC.Text.Length > 0 && txtRua.Text.Length > 0 && txtCodPostal.Text.Length > 0 &&
txtLocalidade.Text.Length > 0 && txtPais.Text.Length > 0)
        {
            tbPessoasRow = (dataSetBiblioteca.PessoasRow)dataTablePessoas.Rows[i];
            string idPessoa = txtIdPessoa.Text.Replace(" ", String.Empty);
            string tipoUtilizador = cbPessoa.SelectedValue.ToString();
            string nome = txtNome.Text.Trim();
            string cc = txtCC.Text.Replace(" ", String.Empty);
            string rua = txtRua.Text.Trim();
            string andar = txtAndar.Text.Replace(" ", String.Empty);
            string letra = txtLetra.Text.Replace(" ", String.Empty);
            string codPostal = txtCodPostal.Text.Replace(" ", String.Empty);
            string localidade = txtLocalidade.Text.Replace(" ", String.Empty);
            string pais = txtPais.Text.Replace(" ", String.Empty);

            if (value == true)
            {
                try
                {
                    if (andar == "")
                        if (letra == "")
                            tableAdapterPessoas.Insert(tipoUtilizador,
Convert.ToInt32(cc), nome, rua, null, null, localidade, codPostal, pais);
                        else
                            tableAdapterPessoas.Insert(tipoUtilizador,
Convert.ToInt32(cc), nome, rua, null, null, letra, localidade, codPostal, pais);
                        else
                            tableAdapterPessoas.Insert(tipoUtilizador,
Convert.ToInt32(cc), nome, rua, null, Convert.ToInt32(andar), letra, localidade,
codPostal, pais);
                }
                catch (Exception error)

```

```

        {
            MessageBox.Show(error.Message.ToString(), "Erro",
MessageBoxButton.OK, MessageBoxIcon.Error);
            return;
        }
        MessageBox.Show("Novo registo inserido com sucesso",
"Confirmação", MessageBoxButton.OK, MessageBoxIcon.Information);
        lblMsg.Visibility = Visibility.Hidden;
        btnCancelar.Visibility = Visibility.Hidden;
        btnEliminar.Visibility = Visibility.Visible;
        lblEmail.Visibility = Visibility.Visible;
        lblContact.Visibility = Visibility.Visible;
        txtEmail.Visibility = Visibility.Visible;
        txtContact.Visibility = Visibility.Visible;
        Bind();
        value = false;
    }
else
{
    try
    {
        tbPessoasRow.Tipo = tipoUtilizador;
        tbPessoasRow.Nome = nome;
        tbPessoasRow.CC = Convert.ToInt32(cc);
        tbPessoasRow.Rua = rua;
        if(andar == "")
            tbPessoasRow.SetAndarNull();
        else
            tbPessoasRow.Andar = Convert.ToInt32(andar);
        if (letra == "")
            tbPessoasRow.SetLetraNull();
        else
            tbPessoasRow.Letra = letra;
        tbPessoasRow.codPostal = codPostal;
        tbPessoasRow.Localidade = localidade;
        tbPessoasRow.Pais = pais;
        tableAdapterPessoas.Update(dataTablePessoas);
    }
    catch (Exception error)
    {
        MessageBox.Show(error.Message.ToString(), "Erro",
MessageBoxButton.OK, MessageBoxIcon.Error);
        return;
    }
    MessageBox.Show("Dados gravados com sucesso!", "Confirmação",
MessageBoxButton.OK, MessageBoxIcon.Information);
    Bind();
}
else
{
    MessageBox.Show("Não é possível inserir registos nulos. Campos
obrigatórios: Categoria de Pessoa, Nome, Cartão Cidadão, Morada, Código de Postal,
Localidade e País", "Erro", MessageBoxButton.OK, MessageBoxIcon.Error);
}
}

private void deleteClick(object sender, RoutedEventArgs e) // elimina um
registro na base de dados
{
    UtilizadoresTableAdapter tableAdapterUtilizadores = new
UtilizadoresTableAdapter();

```

```

        dataSetBiblioteca.UtilizadoresDataTable dataTableUtilizadores;
        dataSetBiblioteca.UtilizadoresRow tbUtilizadoresRow;
        dataTableUtilizadores =
tableAdapterUtilizadores.getDataByUserName(lblUser.Content.ToString());
        tbUtilizadoresRow =
(dataSetBiblioteca.UtilizadoresRow)dataTableUtilizadores[0];
        if (tbUtilizadoresRow.idPessoa == tbPessoasRow.idPessoa)
        {
            MessageBox.Show("Não pode eliminar um utilizador com sessão iniciada.", "Erro", MessageBoxButtons.OK, MessageBoxIcon.Warning);
            return;
        }
        MessageBoxResult confirm = MessageBox.Show("Tem a certeza que pretende eliminar este registo?", "Atenção", MessageBoxButtons.YesNo, MessageBoxIcon.Question);
        if (confirm == MessageBoxResult.Yes)
        {
            try
            {
                tbPessoasRow =
(dataSetBiblioteca.PessoasRow)dataTablePessoas.Rows[i];
                tbPessoasRow.Delete();
                tableAdapterPessoas.Update(dataTablePessoas);
            }
            catch (Exception error)
            {
                MessageBox.Show(error.Message.ToString(), "Erro",
MessageBoxButton.OK, MessageBoxIcon.Warning);
                return;
            }
            MessageBox.Show("Registo eliminado", "Aviso", MessageBoxButtons.OK,
MessageBoxImage.Information);
            Bind();
        }
    }

    private void addEmail(object sender, RoutedEventArgs e) // adiciona um novo email
    {
        if (txtEmail.Text.Length == 0)
            return;
        string idPessoa = txtIdPessoa.Text.Replace(" ", String.Empty);
        string email = txtEmail.Text.Replace(" ", String.Empty);
        try
        {
            tableAdapterEmails.Insert(Convert.ToInt32(idPessoa), email);
        }
        catch (Exception error)
        {
            MessageBox.Show(error.Message.ToString(), "Erro", MessageBoxButton.OK,
MessageBoxImage.Error);
            return;
        }
        MessageBox.Show("Novo email inserido com sucesso", "Confirmação",
MessageBoxButton.OK, MessageBoxIcon.Information);
        txtEmail.Clear();
        Bind();
    }

    private void removeEmail(object sender, RoutedEventArgs e) // remove um email
    {
        if (lvEmail.SelectedItems.Count > 0)
        {

```

```

        string idPessoa = txtIdPessoa.Text.Replace(" ", String.Empty);
        string email = txtEmail.Text.Replace(" ", String.Empty);
        try
        {
            tableAdapterEmails.Delete(Convert.ToInt32(idPessoa), email);
        }
        catch (Exception error)
        {
            MessageBox.Show(error.Message.ToString(), "Erro",
MessageBoxButton.OK, MessageBoxIcon.Error);
            return;
        }
        MessageBox.Show("Email removido com sucesso", "Confirmação",
MessageBoxButton.OK, MessageBoxIcon.Information);
        txtEmail.Clear();
        Bind();
    }
}

private void addContact(object sender, RoutedEventArgs e) // adiciona um novo
número de contacto
{
    if (txtContact.Text.Length == 0)
        return;
    string idPessoa = txtIdPessoa.Text.Replace(" ", String.Empty);
    string contact = txtContact.Text.Replace(" ", String.Empty);
    try
    {
        tableAdapterTelefone.Insert(Convert.ToInt32(idPessoa), contact);
    }
    catch (Exception error)
    {
        MessageBox.Show(error.Message.ToString(), "Erro", MessageBoxButton.OK,
MessageBoxImage.Error);
        return;
    }
    MessageBox.Show("Novo contacto inserido com sucesso", "Confirmação",
MessageBoxButton.OK, MessageBoxIcon.Information);
    txtContact.Clear();
    Bind();
}

private void removeContact(object sender, RoutedEventArgs e) // remove um
número de contacto
{
    if (lvContact.SelectedItems.Count > 0)
    {
        string idPessoa = txtIdPessoa.Text.Replace(" ", String.Empty);
        string contact = txtContact.Text.Replace(" ", String.Empty);
        try
        {
            tableAdapterTelefone.Delete(Convert.ToInt32(idPessoa), contact);
        }
        catch (Exception error)
        {
            MessageBox.Show(error.Message.ToString(), "Erro",
MessageBoxButton.OK, MessageBoxIcon.Error);
            return;
        }
        MessageBox.Show("Contacto removido com sucesso", "Confirmação",
MessageBoxButton.OK, MessageBoxIcon.Information);
        txtContact.Clear();
    }
}

```

```
        Bind();  
    }  
}  
}
```

## Anexo VIII - Janela Utilizadores

```

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;
using System.Data;
using Biblioteca.DataBase;
using Biblioteca.DataBase.dataSetBibliotecaTableAdapters;
using System.Data.SqlClient;
using System.Configuration;

namespace Biblioteca
{
    public partial class Utilizadores : Window
    {
        UtilizadoresTableAdapter tableAdapterUtilizadores = new
        UtilizadoresTableAdapter();
        dataSetBiblioteca.UtilizadoresDataTable dataTableUtilizadores;
        dataSetBiblioteca.UtilizadoresRow tbUtilizadoresRow;

        TiposPessoaTableAdapter tableAdapterTiposPessoa = new
        TiposPessoaTableAdapter();
        dataSetBiblioteca.TiposPessoaDataTable dataTableTiposPessoa;

        PessoasTableAdapter tableAdapterPessoas = new PessoasTableAdapter();
        dataSetBiblioteca.PessoasDataTable dataTablePessas;
        dataSetBiblioteca.PessoasRow tbPessoasRow;

        string userName;
        bool value = false;
        int i = 0;

        public Utilizadores(string _userName)
        {
            InitializeComponent();
            userName = _userName;
            lblUser.Content = userName;
            Bind();
        }

        protected void Bind() // preenche os campos da janela
        {
            i = 0;
            dataTableUtilizadores = tableAdapterUtilizadores.getUtilizadores();
            tbUtilizadoresRow =
            (dataSetBiblioteca.UtilizadoresRow)dataTableUtilizadores.Rows[0];
            lblReg.Content = (i + 1).ToString() + " de " +
            dataTableUtilizadores.Count().ToString();
            txtIdPessoa.Text = tbUtilizadoresRow.idPessoa.ToString();
            txtIdUtilizador.Text = tbUtilizadoresRow.idUtilizador.ToString();
            fillcbTipoUtilizador();
            cbTipoUtilizador.SelectedValue =
            tbUtilizadoresRow.tipoUtilizador.ToString();
            txtUsername.Text = tbUtilizadoresRow.userName.ToString();
        }
    }
}

```

```

        txtPassword.Password = tbUtilizadoresRow.userPassword.ToString();
    }

    protected void fillcbTipoUtilizador() // preenche a lista de valores
TiposPessoa
    {
        dataTableTiposPessoa = tableAdapterTiposPessoa.getTipoPessoa();
        cbTipoUtilizador.ItemsSource = dataTableTiposPessoa;
        cbTipoUtilizador.DisplayMemberPath = "Desig";
        cbTipoUtilizador.SelectedValuePath = "idTipo";
    }

    private void Image_MouseLeftButtonDown_1(object sender, MouseButtonEventArgs e) // regressa à janela Home
    {
        Home home = new Home(userName);
        this.Close();
        home.Show();
    }

    private void btnFirst_Click_1(object sender, RoutedEventArgs e) // avança para
o primeiro registo
    {
        if (value == true)
            return;
        i = 0;
        tbUtilizadoresRow =
(dataSetBiblioteca.UtilizadoresRow)dataTableUtilizadores.Rows[0];
        lblReg.Content = "1 de " + dataTableUtilizadores.Count().ToString();
        txtIdPessoa.Text = tbUtilizadoresRow.idPessoa.ToString();
        txtIdUtilizador.Text = tbUtilizadoresRow.idUtilizador.ToString();
        cbTipoUtilizador.SelectedValue =
tbUtilizadoresRow.tipoUtilizador.ToString();
        txtUsername.Text = tbUtilizadoresRow.userName.ToString();
        txtPassword.Password = tbUtilizadoresRow.userPassword.ToString();
    }

    private void btnPrevious_Click_1(object sender, RoutedEventArgs e) // retorna
ao registo anterior
    {
        if (i <= 0 || value == true)
            return;
        i--;
        tbUtilizadoresRow =
(dataSetBiblioteca.UtilizadoresRow)dataTableUtilizadores.Rows[i];
        lblReg.Content = (i + 1).ToString() + " de " +
dataTableUtilizadores.Count().ToString();
        txtIdPessoa.Text = tbUtilizadoresRow.idPessoa.ToString();
        txtIdUtilizador.Text = tbUtilizadoresRow.idUtilizador.ToString();
        cbTipoUtilizador.SelectedValue =
tbUtilizadoresRow.tipoUtilizador.ToString();
        txtUsername.Text = tbUtilizadoresRow.userName.ToString();
        txtPassword.Password = tbUtilizadoresRow.userPassword.ToString();
    }

    private void btnNext_Click_1(object sender, RoutedEventArgs e) // avança para
o registo seguinte
    {
        if (i >= dataTableUtilizadores.Count() - 1 || value == true)
            return;
        i++;
    }

```

```

        tbUtilizadoresRow =
(dataSetBiblioteca.UtilizadoresRow)dataTableUtilizadores.Rows[i];
        lblReg.Content = (i + 1).ToString() + " de " +
dataTableUtilizadores.Count().ToString();
        txtIdPessoa.Text = tbUtilizadoresRow.idPessoa.ToString();
        txtIdUtilizador.Text = tbUtilizadoresRow.idUtilizador.ToString();
        cbTipoUtilizador.SelectedValue =
tbUtilizadoresRow.tipoUtilizador.ToString();
        txtUsername.Text = tbUtilizadoresRow.userName.ToString();
    }

    private void btnLast_Click_1(object sender, RoutedEventArgs e) // avança para
o último registo
{
    if (value == true)
        return;
    i = dataTableUtilizadores.Count() - 1;
    tbUtilizadoresRow =
(dataSetBiblioteca.UtilizadoresRow)dataTableUtilizadores.Rows[dataTableUtilizadores.Co
unt() - 1];
    lblReg.Content = dataTableUtilizadores.Count().ToString() + " de " +
dataTableUtilizadores.Count().ToString();
    txtIdPessoa.Text = tbUtilizadoresRow.idPessoa.ToString();
    txtIdUtilizador.Text = tbUtilizadoresRow.idUtilizador.ToString();
    cbTipoUtilizador.SelectedValue =
tbUtilizadoresRow.tipoUtilizador.ToString();
    txtUsername.Text = tbUtilizadoresRow.userName.ToString();
    txtPassword.Password = tbUtilizadoresRow.userPassword.ToString();
}

private void insertDataClick(object sender, RoutedEventArgs e) // prepara o
ambiente para a inserção de um novo registo
{
    lblMsg.Visibility = Visibility.Visible;
    btnEliminar.Visibility = Visibility.Hidden;
    btnCancelar.Visibility = Visibility.Visible;
    txtIdPessoa.IsReadOnly = false;
    txtIdUtilizador.IsReadOnly = false;
    txtIdPessoa.Clear();
    txtIdUtilizador.Clear();
    txtPassword.Clear();
    txtUsername.Clear();
    cbTipoUtilizador.SelectedIndex = -1;
    value = true;
}

private void cancelClick(object sender, RoutedEventArgs e) // cancela a
introdução de um novo registo
{
    lblMsg.Visibility = Visibility.Hidden;
    btnEliminar.Visibility = Visibility.Visible;
    btnCancelar.Visibility = Visibility.Hidden;
    txtIdPessoa.IsReadOnly = true;
    txtIdUtilizador.IsReadOnly = true;
    Bind();
    value = false;
}

private void saveDataClick(object sender, RoutedEventArgs e) // guarda as
alterações e os novos registo na base de dados
{

```

```

        if (txtIdPessoa.Text.Length > 0 || txtIdUtilizador.Text.Length > 0 ||
txtPassword.Password.Length > 0 || txtUsername.Text.Length > 0 || 
cbTipoUtilizador.SelectedIndex > -1)
{
    string idPessoa = txtIdPessoa.Text.Replace(" ", String.Empty);
    string idUtilizador = txtIdUtilizador.Text.Replace(" ", String.Empty);
    string utilizador = txtUsername.Text.Replace(" ", String.Empty);
    string tipoUtilizador = cbTipoUtilizador.SelectedValue.ToString();
    string password = txtPassword.Password.Replace(" ", String.Empty);

    if (value == true)
    {
        try
        {
            tableAdapterUtilizadores.Insert(Convert.ToInt32(idPessoa),
Convert.ToInt32(idUtilizador), utilizador, cbTipoUtilizador.SelectedValue.ToString(),
Cripto.sha256(password));
        }
        catch (Exception error)
        {
            MessageBox.Show(error.Message.ToString(), "Erro",
MessageBoxButton.OK, MessageBoxIcon.Error);
            return;
        }
        MessageBox.Show("Novo registo inserido com sucesso",
"Confirmação", MessageBoxButtons.OK, MessageBoxIcon.Information);
        lblMsg.Visibility = Visibility.Hidden;
        btnCancelar.Visibility = Visibility.Hidden;
        txtIdPessoa.IsReadOnly = true;
        txtIdUtilizador.IsReadOnly = true;
        Bind();
        value = false;
    }
    else
    {
        tbUtilizadoresRow =
(dataSetBiblioteca.UtilizadoresRow)dataTableUtilizadores.Rows[i];
        if (idPessoa != tbUtilizadoresRow.idPessoa.ToString() ||
idUtilizador != tbUtilizadoresRow.idUtilizador.ToString() || utilizador !=
tbUtilizadoresRow.userName.ToString() || tipoUtilizador !=
tbUtilizadoresRow.tipoUtilizador.ToString() || password !=
tbUtilizadoresRow.userPassword.ToString())
        {
            try
            {
                dataTablePessas =
tableAdapterPessoas.getDataByIdPessoa(tbUtilizadoresRow.idPessoa);
                tbPessoasRow =
(dataSetBiblioteca.PessoasRow)dataTablePessas.Rows[0];
                tbUtilizadoresRow.userName = utilizador;
                tbUtilizadoresRow.userPassword = Cripto.sha256(password);
                tbUtilizadoresRow.tipoUtilizador = tipoUtilizador;
                tbPessoasRow.Tipo = tipoUtilizador;
                tableAdapterUtilizadores.Update(dataTableUtilizadores);
                tableAdapterPessoas.Update(dataTablePessas);
            }
            catch (Exception error)
            {
                MessageBox.Show(error.Message.ToString(), "Erro",
MessageBoxButton.OK, MessageBoxIcon.Error);
                return;
            }
        }
    }
}

```



## Anexo IX - Janela Categorias Pessoas

```

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;
using System.Data;
using Biblioteca.DataBase;
using Biblioteca.DataBase.dataSetBibliotecaTableAdapters;
using System.Data.SqlClient;
using System.Configuration;

namespace Biblioteca
{
    public partial class tipoPessoa : Window
    {
        TiposPessoaTableAdapter tableAdapterTiposPessoa = new
        TiposPessoaTableAdapter();
        dataSetBiblioteca.TiposPessoaDataTable dataTableTiposPessoa;
        dataSetBiblioteca.TiposPessoaRow tbTiposPessoaRow;

        string userName;
        bool value = false;
        int i = 0;

        public tipoPessoa(string _userName)
        {
            InitializeComponent();
            userName = _userName;
            lblUser.Content = userName;
            Bind();
        }

        protected void Bind() // preenche os campos da janela
        {
            i = 0;
            dataTableTiposPessoa = tableAdapterTiposPessoa.getTipoPessoa();
            tbTiposPessoaRow =
            (dataSetBiblioteca.TiposPessoaRow)dataTableTiposPessoa.Rows[0];
            lblReg.Content = (i + 1).ToString() + " de " +
            dataTableTiposPessoa.Count().ToString();
            txtIdTipo.Text = tbTiposPessoaRow.idTipo.ToString();
            txtDesignacao.Text = tbTiposPessoaRow.Desig.ToString();
        }

        private void Image_MouseLeftButtonDown_1(object sender, MouseButtonEventArgs e) // regressa à janela Home
        {
            Home home = new Home(userName);
            this.Close();
            home.Show();
        }

        private void btnNext_Click_1(object sender, RoutedEventArgs e) // avança para
        o registo seguinte
    }
}

```

```

    {
        if (i >= dataTableTiposPessoa.Count() - 1 || value == true)
            return;
        i++;
        tbTiposPessoaRow =
(dataSetBiblioteca.TiposPessoaRow)dataTableTiposPessoa.Rows[i];
        lblReg.Content = (i + 1).ToString() + " de " +
dataTableTiposPessoa.Count().ToString();
        txtIdTipo.Text = tbTiposPessoaRow.idTipo.ToString();
        txtDesignacao.Text = tbTiposPessoaRow.Desig.ToString();
    }

    private void btnLast_Click_1(object sender, RoutedEventArgs e) // avança para
o último registo
    {
        if (value == true)
            return;
        i = dataTableTiposPessoa.Count() - 1;
        tbTiposPessoaRow =
(dataSetBiblioteca.TiposPessoaRow)dataTableTiposPessoa.Rows[dataTableTiposPessoa.Count
() - 1];
        lblReg.Content = dataTableTiposPessoa.Count.ToString() + " de " +
dataTableTiposPessoa.Count.ToString();
        txtIdTipo.Text = tbTiposPessoaRow.idTipo.ToString();
        txtDesignacao.Text = tbTiposPessoaRow.Desig.ToString();
    }

    private void btnPrevious_Click_1(object sender, RoutedEventArgs e) // retorna
ao registo anterior
    {
        if (i <= 0 || value == true)
            return;
        i--;
        tbTiposPessoaRow =
(dataSetBiblioteca.TiposPessoaRow)dataTableTiposPessoa.Rows[i];
        lblReg.Content = (i + 1).ToString() + " de " +
dataTableTiposPessoa.Count().ToString();
        txtIdTipo.Text = tbTiposPessoaRow.idTipo.ToString();
        txtDesignacao.Text = tbTiposPessoaRow.Desig.ToString();
    }

    private void btnFirst_Click_1(object sender, RoutedEventArgs e) // avança para
o primeiro registo
    {
        if (value == true)
            return;
        i = 0;
        tbTiposPessoaRow =
(dataSetBiblioteca.TiposPessoaRow)dataTableTiposPessoa.Rows[0];
        lblReg.Content = "1 de " + dataTableTiposPessoa.Count.ToString();
        txtIdTipo.Text = tbTiposPessoaRow.idTipo.ToString();
        txtDesignacao.Text = tbTiposPessoaRow.Desig.ToString();
    }

    private void insertDataClick(object sender, RoutedEventArgs e) // prepara o
ambiente para a inserção de um novo registo
    {
        lblMsg.Visibility = Visibility.Visible;
        btnEliminar.Visibility = Visibility.Hidden;
        btnCancelar.Visibility = Visibility.Visible;
        txtDesignacao.Clear();
        txtIdTipo.Clear();
    }

```

```

        value = true;
    }

    private void cancelClick(object sender, RoutedEventArgs e) // cancela a
introdução de um novo registo
    {
        lblMsg.Visibility = Visibility.Hidden;
        btnEliminar.Visibility = Visibility.Visible;
        btnCancelar.Visibility = Visibility.Hidden;
        Bind();
        value = false;
    }

    private void saveDataClick(object sender, RoutedEventArgs e) // guarda as
alterações e os novos registo na base de dados
    {
        if (txtIdTipo.Text.Length > 0 || txtDesignacao.Text.Length > 0)
        {
            string idTipo = txtIdTipo.Text.Replace(" ", String.Empty);
            string desig = txtDesignacao.Text.Trim();
            if (value == true)
            {
                try
                {
                    tableAdapterTiposPessoa.Insert(idTipo, desig);
                }
                catch (Exception error)
                {
                    MessageBox.Show(error.Message.ToString(), "Erro",
MessageBoxButton.OK, MessageBoxIcon.Error);
                    return;
                }
                MessageBox.Show("Novo registo inserido com sucesso",
"Confirmação", MessageBoxButton.OK, MessageBoxIcon.Information);
                lblMsg.Visibility = Visibility.Hidden;
                btnEliminar.Visibility = Visibility.Visible;
                btnCancelar.Visibility = Visibility.Hidden;
                Bind();
                value = false;
            }
            else
            {
                tbTiposPessoaRow =
(dataSetBiblioteca.TiposPessoaRow)dataTableTiposPessoa.Rows[i];
                if (desig != tbTiposPessoaRow.Desig.ToString() || idTipo !=
tbTiposPessoaRow.idTipo.ToString())
                {
                    try
                    {
                        tbTiposPessoaRow.idTipo = idTipo;
                        tbTiposPessoaRow.Desig = desig;
                        tableAdapterTiposPessoa.Update(dataTableTiposPessoa);
                    }
                    catch (Exception error)
                    {
                        MessageBox.Show(error.Message.ToString(), "Erro");
                        return;
                    }
                    MessageBox.Show("Dados gravados com sucesso!", "Confirmação",
MessageBoxButton.OK, MessageBoxIcon.Information);
                    Bind();
                }
            }
        }
    }
}

```



## Anexo XI - Janela Livros

```

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;
using System.Data;
using Biblioteca.DataBase;
using Biblioteca.DataBase.dataSetBibliotecaTableAdapters;
using System.Data.SqlClient;
using System.Configuration;
using Microsoft.Win32;
using System.Net.Cache;

namespace Biblioteca
{
    public partial class Livros : Window
    {

        LivrosTableAdapter tableAdapterLivros = new LivrosTableAdapter();
        dataSetBiblioteca.LivrosDataTable dataTableLivros;
        dataSetBiblioteca.LivrosRow tbLivrosRow;

        EditorasTableAdapter tableAdapterEditoras = new EditorasTableAdapter();
        dataSetBiblioteca.EditorasDataTable dataTableEditoras;

        AutoresTableAdapter tableAdapterAutores = new AutoresTableAdapter();
        dataSetBiblioteca.AutoresDataTable dataTableAutores;
        dataSetBiblioteca.AutoresDataTable dataTableAutoresCB;

        livrosAutoresTableAdapter tableAdapterLivrosAutores = new
        livrosAutoresTableAdapter();

        string userName;
        bool value = false;
        int i = 0;

        public Livros(string _userName)
        {
            InitializeComponent();
            userName = _userName;
            lblUser.Content = userName;
            Bind();
        }

        protected void Bind() // preenche os campos da janela
        {
            i = 0;
            dataTableLivros = tableAdapterLivros.getLivros();
            tbLivrosRow = (dataSetBiblioteca.LivrosRow)dataTableLivros.Rows[0];
            lblReg.Content = (i + 1).ToString() + " de " +
            dataTableLivros.Count().ToString();
            txtISBN.Text = tbLivrosRow.ISBN.ToString();
            txtTitulo.Text = tbLivrosRow.Titulo.ToString();
            txtResumo.Text = tbLivrosRow.Resumo.ToString();
        }
    }
}

```

```

        bindImage();
        fillcbEditora();
        cbEditora.SelectedValue = tbLivrosRow.codEditora.ToString();
        fillListView();
        fillcbAutor();
    }

protected void bindImage() // coloca a imagem da capa do livro na aplicação
{
    if (tbLivrosRow.IsImageNull())
    {
        btnImg.Content = "Imagen";
        imgBookCover.Source = null;
        return;
    }
    btnImg.Content = "Alterar Imagem";
    string folderPath = "\\\\" + 192.168.1.2 + "\\Biblioteca\\bookImages\\";
    folderPath += tbLivrosRow.Imagem.ToString();
    try
    {
        BitmapImage bitMapImage = new BitmapImage();
        bitMapImage.BeginInit();
        bitMapImage.CacheOption = BitmapCacheOption.None;
        bitMapImage.UriCachePolicy = new
RequestCachePolicy(RequestCacheLevel.BypassCache);
        bitMapImage.CacheOption = BitmapCacheOption.OnLoad;
        bitMapImage.CreateOptions = BitmapCreateOptions.IgnoreImageCache;
        bitMapImage.UriSource = new Uri(folderPath);
        bitMapImage.EndInit();
        imgBookCover.Source = bitMapImage;
    }
    catch (Exception error)
    {
        imgBookCover.Source = null;
    }
}

protected void clearValues() // limpa os valores dos campos
{
    txtISBN.Clear();
    txtResumo.Clear();
    txtTitulo.Clear();
    imgBookCover.Source = null;
}

protected void fillcbEditora() // preenche a lista de valores Editoras
{
    dataTableEditoras = tableAdapterEditoras.getEditoras();
    cbEditora.ItemsSource = dataTableEditoras;
    cbEditora.DisplayMemberPath = "Nome";
    cbEditora.SelectedValuePath = "codEditora";
}

protected void fillcbAutor() // preenche a lista de valores Autores
{
    dataTableAutoresCB = tableAdapterAutores.getAutores();
    cbAutores.ItemsSource = dataTableAutoresCB;
    cbAutores.DisplayMemberPath = "Nome";
    cbAutores.SelectedValuePath = "idAutor";
}

protected void fillListView() // preenche os autores do livro

```

```

        {
            dataTableAutores =
tableAdapterAutores.getDataByAutorISBN(tbLivrosRow.ISBN.ToString());
            lvAutores.DataContext = dataTableAutores.DefaultView;
        }

        private void Image_MouseLeftButtonDown_1(object sender, MouseButtonEventArgs e) // regressa à janela Home
        {
            Home home = new Home(userName);
            this.Close();
            home.Show();
        }

        private void btnFirst_Click_1(object sender, RoutedEventArgs e) // avança para o primeiro registo
        {
            if (value == true)
                return;
            i = 0;
            tbLivrosRow = (dataSetBiblioteca.LivrosRow)dataTableLivros.Rows[0];
            lblReg.Content = "1 de " + dataTableLivros.Count().ToString();
            txtISBN.Text = tbLivrosRow.ISBN.ToString();
            txtTitulo.Text = tbLivrosRow.Titulo.ToString();
            cbEditora.SelectedValue = tbLivrosRow.codEditora.ToString();
            fillListView();
            bindImage();
            if (tbLivrosRow.IsResumoNull() == false)
                txtResumo.Text = tbLivrosRow.Resumo.ToString();
        }

        private void btnPrevious_Click_1(object sender, RoutedEventArgs e) // retorna ao registo anterior
        {
            if (i <= 0 || value == true)
                return;
            i--;
            tbLivrosRow = (dataSetBiblioteca.LivrosRow)dataTableLivros.Rows[i];
            lblReg.Content = (i + 1).ToString() + " de " +
dataTableLivros.Count().ToString();
            txtISBN.Text = tbLivrosRow.ISBN.ToString();
            txtTitulo.Text = tbLivrosRow.Titulo.ToString();
            cbEditora.SelectedValue = tbLivrosRow.codEditora.ToString();
            fillListView();
            bindImage();
            if (tbLivrosRow.IsResumoNull() == false)
                txtResumo.Text = tbLivrosRow.Resumo.ToString();
        }

        private void btnNext_Click_1(object sender, RoutedEventArgs e) // avança para o registo seguinte
        {
            if (i >= dataTableLivros.Count() - 1 || value == true)
                return;
            i++;
            tbLivrosRow = (dataSetBiblioteca.LivrosRow)dataTableLivros.Rows[i];
            lblReg.Content = (i + 1).ToString() + " de " +
dataTableLivros.Count().ToString();
            txtISBN.Text = tbLivrosRow.ISBN.ToString();
            txtTitulo.Text = tbLivrosRow.Titulo.ToString();
            cbEditora.SelectedValue = tbLivrosRow.codEditora.ToString();
            fillListView();
        }
    
```

```

        bindImage();
        if (tbLivrosRow.IsResumoNull() == false)
            txtResumo.Text = tbLivrosRow.Resumo.ToString();
    }

    private void btnLast_Click_1(object sender, RoutedEventArgs e) // avança para
o último registo
    {
        if (value == true)
            return;
        i = dataTableLivros.Count() - 1;
        tbLivrosRow =
(dataSetBiblioteca.LivrosRow)dataTableLivros.Rows[dataTableLivros.Count() - 1];
        lblReg.Content = dataTableLivros.Count().ToString() + " de " +
dataTableLivros.Count().ToString();
        txtISBN.Text = tbLivrosRow.ISBN.ToString();
        txtTitulo.Text = tbLivrosRow.Titulo.ToString();
        cbEditora.SelectedValue = tbLivrosRow.codEditora.ToString();
        fillListView();
        bindImage();
        if (tbLivrosRow.IsResumoNull() == false)
            txtResumo.Text = tbLivrosRow.Resumo.ToString();
    }

    private void insertDataClick(object sender, RoutedEventArgs e) // prepara o
ambiente para a inserção de um novo registo
    {
        lblMsg.Visibility = Visibility.Visible;
        btnEliminar.Visibility = Visibility.Hidden;
        btnCancelar.Visibility = Visibility.Visible;
        btnPDF.Visibility = Visibility.Hidden;
        clearValues();
        cbEditora.SelectedIndex = -1;
        lvAutores.DataContext = null;
        btnNewAuthor.IsEnabled = false;
        btnRemoveAuthor.IsEnabled = false;
        btnChangePDF.IsEnabled = false;
        btnImg.IsEnabled = false;
        value = true;
    }

    private void cancelClick(object sender, RoutedEventArgs e) // cancela a
introdução de um novo registo
    {
        lblMsg.Visibility = Visibility.Hidden;
        btnEliminar.Visibility = Visibility.Visible;
        btnCancelar.Visibility = Visibility.Hidden;
        btnPDF.Visibility = Visibility.Visible;
        btnNewAuthor.IsEnabled = true;
        btnRemoveAuthor.IsEnabled = true;
        btnChangePDF.IsEnabled = true;
        btnImg.IsEnabled = true;
        Bind();
        value = false;
    }

    private void saveDataClick(object sender, RoutedEventArgs e) // guarda as
alterações e os novos registo na base de dados
    {
        if (txtISBN.Text.Length > 0 && cbEditora.SelectedIndex > -1 &&
txtTitulo.Text.Length > 0)
        {

```

```

        string isbn = txtISBN.Text.Replace(" ", String.Empty);
        string editora = cbEditora.SelectedValue.ToString();
        string titulo = txtTitulo.Text.Trim();
        string resumo = txtResumo.Text.Trim();
        if (resumo.Length == 0)
            resumo = "Resumo não disponivel";
        if (value == true)
        {
            try
            {
                tableAdapterLivros.Insert(isbn, titulo,
Convert.ToInt32(editora), null, resumo, null, null);
            }
            catch (Exception error)
            {
                MessageBox.Show(error.Message.ToString(), "Erro",
MessageBoxButton.OK, MessageBoxIcon.Error);
                return;
            }
        }
        else
        {
            try
            {
                tbLivrosRow.ISBN = isbn;
                tbLivrosRow.Titulo = titulo;
                tbLivrosRow.codEditora = Convert.ToInt32(editora);
                tbLivrosRow.Resumo = resumo;
                tableAdapterLivros.Update(dataTableLivros);
            }
            catch (Exception error)
            {
                MessageBox.Show(error.Message.ToString(), "Erro",
MessageBoxButton.OK, MessageBoxIcon.Error);
                return;
            }
        }
        MessageBox.Show("Dados gravados com sucesso!", "Confirmação",
MessageBoxButton.OK, MessageBoxIcon.Information);
        lblMsg.Visibility = Visibility.Hidden;
        btnEliminar.Visibility = Visibility.Visible;
        btnCancelar.Visibility = Visibility.Hidden;
        btnPDF.Visibility = Visibility.Visible;
        btnNewAuthor.IsEnabled = true;
        btnRemoveAuthor.IsEnabled = true;
        btnChangePDF.IsEnabled = true;
        btnImg.IsEnabled = true;
        Bind();
        value = false;
    }
    else
    {
        MessageBox.Show("Não é possível inserir registos nulos. Campos
obrigatórios: ISBN, Editora, Titulo", "Erro", MessageBoxButton.OK,
MessageBoxImage.Error);
    }
}

private void deleteClick(object sender, RoutedEventArgs e) // elimina um
registro na base de dados
{
    if (lvAutores.Items.Count > 0)

```

```

        {
            MessageBox.Show("Remova os autores associados a este livro.", "Erro",
                MessageBoxButtons.OK, MessageBoxIcon.Error);
            return;
        }
        MessageBoxResult confirm = MessageBox.Show("Tem a certeza que pretende
    eliminar este registo?", "Atenção", MessageBoxButtons.YesNo, MessageBoxIcon.Question);
        if (confirm == MessageBoxResult.Yes)
        {
            try
            {
                if (tbLivrosRow.IsImageNull() == false)
                {
                    string imageFolderPath =
                    "\\\\" + 192.168.1.2 + "\\Biblioteca\\bookImages\\";
                    string imageFilePath = imageFolderPath +
                    tbLivrosRow.Imagem.ToString().Trim();
                    if (System.IO.File.Exists(imageFilePath))
                    {
                        try
                        {
                            System.IO.File.Delete(imageFilePath);
                        }
                        catch (System.IO.IOException error)
                        {
                            MessageBox.Show("Erro ao eliminar imagem, por favor
    contacte o Administrador de Sistema", "Erro", MessageBoxButtons.OK,
                                MessageBoxIcon.Warning);
                            return;
                        }
                    }
                }
                if (tbLivrosRow.IsPDFNull() == false)
                {
                    string booksFolderPath =
                    "\\\\" + 192.168.1.2 + "\\Biblioteca\\Books\\";
                    string booksFilePath = booksFolderPath +
                    tbLivrosRow.PDF.ToString().Trim();
                    if (System.IO.File.Exists(booksFilePath))
                    {
                        try
                        {
                            System.IO.File.Delete(booksFilePath);
                        }
                        catch (System.IO.IOException error)
                        {
                            MessageBox.Show("Erro ao eliminar livro, por favor
    contacte o Administrador de Sistema", "Erro", MessageBoxButtons.OK,
                                MessageBoxIcon.Warning);
                            return;
                        }
                    }
                }
                tbLivrosRow =
                (dataSetBiblioteca.LivrosRow)dataTableLivros.Rows[i];
                tbLivrosRow.Delete();
                tableAdapterLivros.Update(dataTableLivros);
            }
            catch (Exception error)
            {

```

```

        MessageBox.Show(error.Message.ToString(), "Erro",
MessageBoxButton.OK, MessageBoxIcon.Warning);
        return;
    }
    MessageBox.Show("Registo eliminado", "Aviso", MessageBoxButton.OK,
MessageBoxImage.Information);
    Bind();
}
}

private void changeCover(object sender, RoutedEventArgs e) // altera a imagem
da capa do livro
{
    MessageBoxResult confirm = MessageBox.Show("Esta operação elimina a imagem
da capa anterior caso exista. Pretende continuar?", "Atenção", MessageBoxButton.YesNo,
MessageBoxImage.Question);
    if (confirm == MessageBoxResult.No)
        return;
    OpenFileDialog CxDialog = new OpenFileDialog();
    string folderPath = "\\\\" + 192.168.1.2 + "\\Biblioteca\\bookImages\\";
    CxDialog.Title = "Seleccione a capa pretendida";
    CxDialog.Filter = "JPEG (*.jpg)|*.jpg";
    CxDialog.FilterIndex = 1;
    CxDialog.CheckFileExists = true;
    bool? myResult;
    myResult = CxDialog.ShowDialog();
    if (myResult != null && myResult == true)
    {
        if (tbLivrosRow.ImgagemNull())
        {
            tbLivrosRow.Imgagem = tbLivrosRow.ISBN.ToString() + ".jpg";
            try
            {
                tableAdapterLivros.Update(dataTableLivros);
            }
            catch (Exception error)
            {
                MessageBox.Show(error.Message.ToString(), "Erro",
MessageBoxButton.OK, MessageBoxIcon.Error);
                return;
            }
        }
        string filePath = folderPath + tbLivrosRow.Imgagem.ToString().Trim();
        System.IO.File.Copy(CxDialog.FileName, filePath, true);
        imgBookCover.Source = null;
        MessageBox.Show("Imagen inserida com sucesso.", "Aviso",
MessageBoxButton.OK, MessageBoxIcon.Information);
        Bind();
    }
}

private void openPDF(object sender, RoutedEventArgs e) // abre o ficheiro PDF
do livro
{
    try
    {
        System.Diagnostics.Process process = new System.Diagnostics.Process();
        string path = "\\\\" + 192.168.1.2 + "\\Biblioteca\\Books\\" +
tbLivrosRow.PDF.ToString();
        Uri pdf = new Uri(path, UriKind.RelativeOrAbsolute);
        process.StartInfo.FileName = pdf.LocalPath;
        process.Start();
    }
}

```

```

        process.WaitForExit();
    }
    catch (Exception error)
    {
        MessageBox.Show("Não foi possível visualizar o livro.", "Erro",
    MessageBoxButtons.OK, MessageBoxIcon.Warning);
    }
}

private void changePDF(object sender, RoutedEventArgs e) // altera o ficheiro
PDF do livro
{
    MessageBoxResult confirm = MessageBox.Show("Esta operação elimina PDF
anterior caso exista. Pretende continuar?", "Atenção", MessageBoxButtons.YesNo,
MessageBoxImage.Question);
    if (confirm == MessageBoxResult.No)
        return;
    OpenFileDialog CxDialog = new OpenFileDialog();
    string folderPath = "\\\\" + 192.168.1.2 + "\\Biblioteca\\Books\\";
    CxDialog.Title = "Seleccione o PDF pretendido";
    CxDialog.Filter = "Adobe PDF Files (*.pdf)|*.pdf";
    CxDialog.FilterIndex = 1;
    CxDialog.CheckFileExists = true;
    bool? myResult;
    myResult = CxDialog.ShowDialog();
    if (myResult != null && myResult == true)
    {
        if (tbLivrosRow.IsPDFNull())
        {
            tbLivrosRow.PDF = tbLivrosRow.ISBN.ToString() + ".pdf";
            try
            {
                tableAdapterLivros.Update(dataTableLivros);
            }
            catch (Exception error)
            {
                MessageBox.Show(error.Message.ToString(), "Erro",
    MessageBoxButtons.OK, MessageBoxIcon.Error);
                return;
            }
        }
        string filePath = folderPath + tbLivrosRow.PDF.ToString().Trim();
        System.IO.File.Copy(CxDialog.FileName, filePath, true);
        MessageBox.Show("PDF inserido com sucesso.", "Aviso",
    MessageBoxButtons.OK, MessageBoxIcon.Information);
        Bind();
    }
}

private void addAuthor(object sender, RoutedEventArgs e) // adiciona um autor
ao livro
{
    if (cbAutores.SelectedIndex < 0)
        return;
    try
    {

tableAdapterLivrosAutores.Insert(Convert.ToInt32(cbAutores.SelectedValue.ToString()),
txtISBN.Text.ToString());
    }
    catch (Exception error)
    {

```

```
        MessageBox.Show(error.Message.ToString(), "Erro", MessageBoxButtons.OK,
MessageBoxImage.Error);
        return;
    }
    MessageBox.Show("Autor inserido com sucesso", "Confirmação",
MessageBoxButton.OK, MessageBoxIcon.Information);
    Bind();
}

private void removeAuthor(object sender, RoutedEventArgs e) // remove um autor
do livro
{
    if (lvAutores.SelectedItems.Count > 0)
    {
        try
        {

tableAdapterLivrosAutores.Delete(Convert.ToInt32(cbAutores.SelectedValue.ToString()),
txtISBN.Text.ToString());
        }
        catch (Exception error)
        {
            MessageBox.Show(error.Message.ToString(), "Erro",
MessageBoxButton.OK, MessageBoxIcon.Error);
            return;
        }
        MessageBox.Show("Autor removido com sucesso", "Confirmação",
MessageBoxButton.OK, MessageBoxIcon.Information);
        Bind();
    }
}
}
```

## Anexo XII - Janela Editoras

```

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;
using System.Data;
using Biblioteca.DataBase;
using Biblioteca.DataBase.dataSetBibliotecaTableAdapters;
using System.Data.SqlClient;
using System.Configuration;

namespace Biblioteca
{
    public partial class Editoras : Window
    {

        EditorasTableAdapter tableAdapterEditoras = new EditorasTableAdapter();
        dataSetBiblioteca.EditorasDataTable dataTableEditoras;
        dataSetBiblioteca.EditorasRow tbEditorasRow;

        string userName;
        bool value = false;
        int i = 0;

        public Editoras(string _userName)
        {
            InitializeComponent();
            userName = _userName;
            lblUser.Content = userName;
            Bind();
        }

        private void Image_MouseLeftButtonDown_1(object sender, MouseButtonEventArgs e) // regressa à janela Home
        {
            Home home = new Home(userName);
            this.Close();
            home.Show();
        }

        protected void Bind() // preenche os campos da janela
        {
            i = 0;
            dataTableEditoras = tableAdapterEditoras.getEditoras();
            tbEditorasRow = (dataSetBiblioteca.EditorasRow)dataTableEditoras.Rows[0];
            lblReg.Content = (i + 1).ToString() + " de " +
                dataTableEditoras.Count().ToString();
            txtIdEditora.Text = tbEditorasRow.codEditora.ToString();
            if (tbEditorasRow.IsNomeNull() == false)
                txtNome.Text = tbEditorasRow.Nome.ToString();
            if (tbEditorasRow.IsRuaNull() == false)
                txtRua.Text = tbEditorasRow.Rua.ToString();
            if (tbEditorasRow.IsAndarNull() == false)
                txtAndar.Text = tbEditorasRow.Andar.ToString();
        }
    }
}

```

```

        if (tbEditorasRow.IsLetraNull() == false)
            txtLetra.Text = tbEditorasRow.Letra.ToString();
        if (tbEditorasRow.IscodPostalNull() == false)
            txtCodPostal.Text = tbEditorasRow.codPostal.ToString();
        if (tbEditorasRow.IsLocalidadeNull() == false)
            txtLocalidade.Text = tbEditorasRow.Localidade.ToString();
        if (tbEditorasRow.IsPaisNull() == false)
            txtPais.Text = tbEditorasRow.Pais.ToString();
    }

    protected void clearValues() // limpa os valores dos campos
    {
        txtNome.Clear();
        txtRua.Clear();
        txtAndar.Clear();
        txtLetra.Clear();
        txtCodPostal.Clear();
        txtLocalidade.Clear();
        txtPais.Clear();
    }

    private void btnFirst_Click_1(object sender, RoutedEventArgs e) // avança para
o primeiro registo
    {
        if (value == true)
            return;
        i = 0;
        tbEditorasRow = (dataSetBiblioteca.EditorasRow)dataTableEditoras.Rows[0];
        lblReg.Content = "1 de " + dataTableEditoras.Count().ToString();
        txtIdEditora.Text = tbEditorasRow.codEditora.ToString();
        clearValues();
        if (tbEditorasRow.IsNomeNull() == false)
            txtNome.Text = tbEditorasRow.Nome.ToString();
        if (tbEditorasRow.IsRuaNull() == false)
            txtRua.Text = tbEditorasRow.Rua.ToString();
        if (tbEditorasRow.IsAndarNull() == false)
            txtAndar.Text = tbEditorasRow.Andar.ToString();
        if (tbEditorasRow.IsLetraNull() == false)
            txtLetra.Text = tbEditorasRow.Letra.ToString();
        if (tbEditorasRow.IscodPostalNull() == false)
            txtCodPostal.Text = tbEditorasRow.codPostal.ToString();
        if (tbEditorasRow.IsLocalidadeNull() == false)
            txtLocalidade.Text = tbEditorasRow.Localidade.ToString();
        if (tbEditorasRow.IsPaisNull() == false)
            txtPais.Text = tbEditorasRow.Pais.ToString();
    }

    private void btnPrevious_Click_1(object sender, RoutedEventArgs e) // retorna
ao registo anterior
    {
        if (i <= 0 || value == true)
            return;
        i--;
        tbEditorasRow = (dataSetBiblioteca.EditorasRow)dataTableEditoras.Rows[i];
        lblReg.Content = (i + 1).ToString() + " de " +
dataTableEditoras.Count().ToString();
        txtIdEditora.Text = tbEditorasRow.codEditora.ToString();
        clearValues();
        if (tbEditorasRow.IsNomeNull() == false)
            txtNome.Text = tbEditorasRow.Nome.ToString();
        if (tbEditorasRow.IsRuaNull() == false)
            txtRua.Text = tbEditorasRow.Rua.ToString();
    }

```

```

        if (tbEditorasRow.IsAndarNull() == false)
            txtAndar.Text = tbEditorasRow.Andar.ToString();
        if (tbEditorasRow.IsLetraNull() == false)
            txtLetra.Text = tbEditorasRow.Letra.ToString();
        if (tbEditorasRow.IscodPostalNull() == false)
            txtCodPostal.Text = tbEditorasRow.codPostal.ToString();
        if (tbEditorasRow.IsLocalidadeNull() == false)
            txtLocalidade.Text = tbEditorasRow.Localidade.ToString();
        if (tbEditorasRow.IsPaisNull() == false)
            txtPais.Text = tbEditorasRow.Pais.ToString();
    }

    private void btnNext_Click_1(object sender, RoutedEventArgs e) // avança para
o registo seguinte
{
    if (i >= dataTableEditoras.Count() - 1 || value == true)
        return;
    i++;
    tbEditorasRow = (dataSetBiblioteca.EditorasRow)dataTableEditoras.Rows[i];
    lblReg.Content = (i + 1).ToString() + " de " +
dataTableEditoras.Count().ToString();
    txtIdEditora.Text = tbEditorasRow.codEditora.ToString();
    clearValues();
    if (tbEditorasRow.IsNomeNull() == false)
        txtNome.Text = tbEditorasRow.Nome.ToString();
    if (tbEditorasRow.IsRuaNull() == false)
        txtRua.Text = tbEditorasRow.Rua.ToString();
    if (tbEditorasRow.IsAndarNull() == false)
        txtAndar.Text = tbEditorasRow.Andar.ToString();
    if (tbEditorasRow.IsLetraNull() == false)
        txtLetra.Text = tbEditorasRow.Letra.ToString();
    if (tbEditorasRow.IscodPostalNull() == false)
        txtCodPostal.Text = tbEditorasRow.codPostal.ToString();
    if (tbEditorasRow.IsLocalidadeNull() == false)
        txtLocalidade.Text = tbEditorasRow.Localidade.ToString();
    if (tbEditorasRow.IsPaisNull() == false)
        txtPais.Text = tbEditorasRow.Pais.ToString();
}

private void btnLast_Click_1(object sender, RoutedEventArgs e) // avança para
o último registo
{
    if (value == true)
        return;
    i = dataTableEditoras.Count() - 1;
    tbEditorasRow =
(dataSetBiblioteca.EditorasRow)dataTableEditoras.Rows[dataTableEditoras.Count() - 1];
    lblReg.Content = dataTableEditoras.Count().ToString() + " de " +
dataTableEditoras.Count().ToString();
    txtIdEditora.Text = tbEditorasRow.codEditora.ToString();
    clearValues();
    if (tbEditorasRow.IsNomeNull() == false)
        txtNome.Text = tbEditorasRow.Nome.ToString();
    if (tbEditorasRow.IsRuaNull() == false)
        txtRua.Text = tbEditorasRow.Rua.ToString();
    if (tbEditorasRow.IsAndarNull() == false)
        txtAndar.Text = tbEditorasRow.Andar.ToString();
    if (tbEditorasRow.IsLetraNull() == false)
        txtLetra.Text = tbEditorasRow.Letra.ToString();
    if (tbEditorasRow.IscodPostalNull() == false)
        txtCodPostal.Text = tbEditorasRow.codPostal.ToString();
    if (tbEditorasRow.IsLocalidadeNull() == false)

```

```

        txtLocalidade.Text = tbEditorasRow.Localidade.ToString();
        if (tbEditorasRow.IsPaisNull() == false)
            txtPais.Text = tbEditorasRow.Pais.ToString();
    }

    private void insertDataClick(object sender, RoutedEventArgs e) // prepara o
ambiente para a inserção de um novo registo
    {
        lblMsg.Visibility = Visibility.Visible;
        btnEliminar.Visibility = Visibility.Hidden;
        btnCancelar.Visibility = Visibility.Visible;
        clearValues();
        txtIdEditora.Clear();
        value = true;
    }

    private void cancelClick(object sender, RoutedEventArgs e) // cancela a
introdução de um novo registo
    {
        lblMsg.Visibility = Visibility.Hidden;
        btnEliminar.Visibility = Visibility.Visible;
        btnCancelar.Visibility = Visibility.Hidden;
        Bind();
        value = false;
    }

    private void saveDataClick(object sender, RoutedEventArgs e) // guarda as
alterações e os novos registo na base de dados
    {
        if (txtNome.Text.Length > 0 && txtRua.Text.Length > 0 &&
txtCodPostal.Text.Length > 0 && txtLocalidade.Text.Length > 0 && txtPais.Text.Length >
0)
        {
            tbEditorasRow =
(dataSetBiblioteca.EditorasRow)dataTableEditoras.Rows[i];
            string codEditora = txtIdEditora.Text.Replace(" ", String.Empty);
            string nome = txtNome.Text.Trim();
            string rua = txtRua.Text.Trim();
            string andar = txtAndar.Text.Replace(" ", String.Empty);
            string letra = txtLetra.Text.Replace(" ", String.Empty);
            string codPostal = txtCodPostal.Text.Replace(" ", String.Empty);
            string localidade = txtLocalidade.Text.Trim();
            string pais = txtPais.Text.Trim();

            if (value == true)
            {
                try
                {
                    if (andar == "")
                        if (letra == "")
                            tableAdapterEditoras.Insert(nome, rua, null, null,
localidade, codPostal, pais);
                        else
                            tableAdapterEditoras.Insert(nome, rua, null, letra,
localidade, codPostal, pais);
                    else
                        tableAdapterEditoras.Insert(nome, rua,
Convert.ToInt32(andar), letra, localidade, codPostal, pais);
                }
                catch (Exception error)
                {

```

```

        MessageBox.Show(error.Message.ToString(), "Erro",
MessageBoxButton.OK, MessageBoxIcon.Error);
        return;
    }
    MessageBox.Show("Novo registo inserido com sucesso",
"Confirmação", MessageBoxButton.OK, MessageBoxIcon.Information);
lblMsg.Visibility = Visibility.Hidden;
btnCancelar.Visibility = Visibility.Hidden;
btnEliminar.Visibility = Visibility.Visible;
Bind();
value = false;
return;
}
else
{
try
{
    tbEditorasRow.Nome = nome;
    tbEditorasRow.Rua = rua;
    if (andar == "")
        tbEditorasRow.SetAndarNull();
    else
        tbEditorasRow.Andar = Convert.ToInt32(andar);
    if (letra == "")
        tbEditorasRow.SetLetraNull();
    else
        tbEditorasRow.Letra = letra;
    tbEditorasRow.codPostal = codPostal;
    tbEditorasRow.Localidade = localidade;
    tbEditorasRow.Pais = pais;
    tableAdapterEditoras.Update(dataTableEditoras);
}
catch (Exception error)
{
    MessageBox.Show(error.Message.ToString(), "Erro",
MessageBoxButton.OK, MessageBoxIcon.Error);
    return;
}
}
MessageBox.Show("Dados gravados com sucesso!", "Confirmação",
MessageBoxButton.OK, MessageBoxIcon.Information);
Bind();
}
else
{
    MessageBox.Show("Não é possível inserir registos nulos. Campos
obrigatórios: Categoria de Pessoa, Nome, Cartão Cidadão, Morada, Código de Postal,
Localidade e País", "Erro", MessageBoxButton.OK, MessageBoxIcon.Error);
}
}

private void deleteClick(object sender, RoutedEventArgs e) // elimina um
registro na base de dados
{
    LivrosTableAdapter tableAdapterLivros = new LivrosTableAdapter();

    if
(tableAdapterLivros.getDataByCodEditora(Convert.ToInt32(txtIdEditora.Text)).Rows.Count
> 0)
    {
        MessageBox.Show("Não pode eliminar uma editora associada a livros.",
"Erro", MessageBoxButton.OK, MessageBoxIcon.Warning);
    }
}

```

```
        return;
    }
    MessageBoxResult confirm = MessageBox.Show("Tem a certeza que pretende
eliminar este registo?", "Atenção", MessageBoxButtons.YesNo, MessageBoxIcon.Question);
    if (confirm == MessageBoxResult.Yes)
    {
        try
        {
            tbEditorasRow =
(dataSetBiblioteca.EditorasRow)dataTableEditoras.Rows[i];
            tbEditorasRow.Delete();
            tableAdapterEditoras.Update(dataTableEditoras);
        }
        catch (Exception error)
        {
            MessageBox.Show(error.Message.ToString(), "Erro",
MessageBoxButton.OK, MessageBoxIcon.Warning);
            return;
        }
        MessageBox.Show("Registo eliminado", "Aviso", MessageBoxButtons.OK,
MessageBoxImage.Information);
        Bind();
    }
}
}
```

## Anexo XIII - Janela Editoras

```

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;
using System.Data;
using Biblioteca.DataBase;
using Biblioteca.DataBase.dataSetBibliotecaTableAdapters;
using System.Data.SqlClient;
using System.Configuration;

namespace Biblioteca
{
    public partial class Autores : Window
    {
        AutoresTableAdapter tableAdapterAutores = new AutoresTableAdapter();
        dataSetBiblioteca.AutoresDataTable dataTableAutores;
        dataSetBiblioteca.AutoresRow tbAutoresRow;

        string userName;
        bool value = false;
        int i = 0;

        public Autores(string _userName)
        {
            InitializeComponent();
            userName = _userName;
            lblUser.Content = userName;
            Bind();
        }

        protected void Bind() // preenche os campos da janela
        {
            i = 0;
            dataTableAutores = tableAdapterAutores.getAutores();
            tbAutoresRow = (dataSetBiblioteca.AutoresRow)dataTableAutores.Rows[0];
            lblReg.Content = (i + 1).ToString() + " de " +
            dataTableAutores.Count().ToString();
            txtIdAutor.Text = tbAutoresRow.idAutor.ToString();
            txtNome.Text = tbAutoresRow.Nome.ToString();
        }

        private void Image_MouseLeftButtonDown_1(object sender, MouseButtonEventArgs e) // regressa à janela Home
        {
            Home home = new Home(userName);
            this.Close();
            home.Show();
        }

        private void btnNext_Click_1(object sender, RoutedEventArgs e) // avança para o registo seguinte
        {
            if (i >= dataTableAutores.Count() - 1 || value == true)

```

```

        return;
    i++;
    tbAutoresRow = (dataSetBiblioteca.AutoresRow)dataTableAutores.Rows[i];
    lblReg.Content = (i + 1).ToString() + " de " +
    dataTableAutores.Count().ToString();
    txtIdAutor.Text = tbAutoresRow.idAutor.ToString();
    txtNome.Text = tbAutoresRow.Nome.ToString();

}

private void btnLast_Click_1(object sender, RoutedEventArgs e) // avança para
o primeiro registo
{
    if (value == true)
        return;
    i = dataTableAutores.Count() - 1;
    tbAutoresRow =
(dataSetBiblioteca.AutoresRow)dataTableAutores.Rows[dataTableAutores.Count() - 1];
    lblReg.Content = (i + 1).ToString() + " de " +
dataTableAutores.Count().ToString();
    txtIdAutor.Text = tbAutoresRow.idAutor.ToString();
    txtNome.Text = tbAutoresRow.Nome.ToString();
}

private void btnPrevious_Click_1(object sender, RoutedEventArgs e) // retorna
ao registo anterior
{
    if (i <= 0 || value == true)
        return;
    i--;
    tbAutoresRow = (dataSetBiblioteca.AutoresRow)dataTableAutores.Rows[i];
    lblReg.Content = (i + 1).ToString() + " de " +
dataTableAutores.Count().ToString();
    txtIdAutor.Text = tbAutoresRow.idAutor.ToString();
    txtNome.Text = tbAutoresRow.Nome.ToString();
}

private void btnFirst_Click_1(object sender, RoutedEventArgs e) // avança para
o primeiro registo
{
    if (value == true)
        return;
    i = 0;
    tbAutoresRow = (dataSetBiblioteca.AutoresRow)dataTableAutores.Rows[0];
    lblReg.Content = "1 de " + dataTableAutores.Count().ToString();
    txtIdAutor.Text = tbAutoresRow.idAutor.ToString();
    txtNome.Text = tbAutoresRow.Nome.ToString();
}

private void insertDataClick(object sender, RoutedEventArgs e) // prepara o
ambiente para a inserção de um novo registo
{
    lblMsg.Visibility = Visibility.Visible;
    btnEliminar.Visibility = Visibility.Hidden;
    btnCancelar.Visibility = Visibility.Visible;
    txtNome.Clear();
    txtIdAutor.Clear();
    value = true;
}

private void cancelClick(object sender, RoutedEventArgs e) // cancela a
introdução de um novo registo

```

```

{
    lblMsg.Visibility = Visibility.Hidden;
    btnEliminar.Visibility = Visibility.Visible;
    btnCancelar.Visibility = Visibility.Hidden;
    Bind();
    value = false;
}

private void saveDataClick(object sender, RoutedEventArgs e) // guarda as
alterações e os novos registo na base de dados
{
    if (txtNome.Text.Length > 0)
    {
        string nome = txtNome.Text.Trim();
        if (value == true)
        {
            try
            {
                tableAdapterAutores.Insert(nome);
            }
            catch (Exception error)
            {
                MessageBox.Show(error.Message.ToString(), "Erro",
MessageBoxButton.OK, MessageBoxIcon.Error);
                return;
            }
            MessageBox.Show("Novo registo inserido com sucesso",
"Confirmação", MessageBoxButton.OK, MessageBoxIcon.Information);
            lblMsg.Visibility = Visibility.Hidden;
            btnEliminar.Visibility = Visibility.Visible;
            btnCancelar.Visibility = Visibility.Hidden;
            Bind();
            value = false;
        }
        else
        {
            tbAutoresRow =
(dataSetBiblioteca.AutoresRow)dataTableAutores.Rows[i];
            if (nome != tbAutoresRow.Nome.ToString())
            {
                try
                {
                    tbAutoresRow.Nome = nome;
                    tableAdapterAutores.Update(dataTableAutores);
                }
                catch (Exception error)
                {
                    MessageBox.Show(error.Message.ToString(), "Erro");
                    return;
                }
                MessageBox.Show("Dados gravados com sucesso!", "Confirmação",
MessageBoxButton.OK, MessageBoxIcon.Information);
                Bind();
            }
        }
    }
    else
    {
        MessageBox.Show("Não é possível inserir registo nulos", "Erro",
MessageBoxButton.OK, MessageBoxIcon.Error);
    }
}

```

```

    private void deleteClick(object sender, RoutedEventArgs e) // elimina um
registro na base de dados
    {
        livrosAutoresTableAdapter tableAdapterLivrosAutores = new
livrosAutoresTableAdapter();
        if
(tableAdapterLivrosAutores.getDataByIdAutor(Convert.ToInt32(txtIdAutor.Text)).Rows.Cou
nt > 0)
        {
            MessageBox.Show("Não pode eliminar um autor associado a livros.",
"Erro", MessageBoxButton.OK, MessageBoxIcon.Warning);
            return;
        }
        MessageBoxResult confirm = MessageBox.Show("Tem a certeza que pretende
eliminar este registo?", "Atenção", MessageBoxButton.YesNo, MessageBoxIcon.Question);
        if (confirm == MessageBoxResult.Yes)
        {
            try
            {
                tbAutoresRow =
(dataSetBiblioteca.AutoresRow)dataTableAutores.Rows[i];
                tbAutoresRow.Delete();
                tableAdapterAutores.Update(dataTableAutores);
            }
            catch (Exception error)
            {
                MessageBox.Show(error.Message.ToString(), "Erro",
MessageBoxButton.OK, MessageBoxIcon.Warning);
                return;
            }
            MessageBox.Show("Registo eliminado", "Aviso", MessageBoxButton.OK,
MessageBoxImage.Information);
            Bind();
        }
    }
}

```

