

Relatório da Etapa 3:

ViewModel e Fim do Projeto

Engenharia Informática

Laboratório de Planeamento e Desenvolvimento de Software

Luís Filipe Leite Barbosa

David Patrício Luna

**Autores**

Bruno Carvalho - al73235

José Silva - al73706

Ricardo Inácio - al73246

Tiago Fernandes - al73701

Vila Real, 2022

**Resumo**

Este projeto consiste no planeamento e desenvolvimento de duas aplicações, para duas plataformas diferentes (Windows e Android), usando respetivamente, as tecnologias **WPF** e **Xamarin**, utilizando a arquitetura de software **MVVM** (Model, View, View-Model), que permite de uma forma eficiente, a distribuição de tarefas por equipas, e a reutilização e reaproveitamento de código já criado. Com esta arquitetura, a parte “lógica” do programa separa-se da interface gráfica, de modo que tanto a aplicação Android (Xamarin) e a “Desktop” (Windows Plataform Foundation), apenas divirjam neste último aspeto, tendo de ser adaptado a cada um dos respetivos dispositivos onde estes sistemas operativos correm. Devemos referir que as tecnologias utilizadas, como ambiente de desenvolvimento, será o Visual Studio 2022, C#, e .NET 6.0. O objetivo principal deste projeto, é que as aplicações, no final da sua conceção, sejam colaborativas, e que permitam gerir listas de produtos reais (de lojas físicas, por exemplo).

Índice

[1. Introdução 1](#_Toc106488656)

[2. Enquadramento Teórico 1](#_Toc106488657)

[3. Objetivos da Etapa do Trabalho Prático 3](#_Toc106488658)

[4. Desenvolvimento 4](#_Toc106488659)

[WPF 4](#_Toc106488660)

[ViewModel.cs 4](#_Toc106488661)

[Login.xaml.cs 18](#_Toc106488662)

[Registar.xaml.cs 20](#_Toc106488663)

[EditarConta.xaml.cs 22](#_Toc106488664)

[CriarLista.xaml.cs 24](#_Toc106488665)

[JanelaProduto.xaml.cs 31](#_Toc106488666)

[MainWindow.xaml.cs 35](#_Toc106488667)

[Xamarain 43](#_Toc106488668)

[RegisterPage.xaml.cs 43](#_Toc106488669)

[LoginPage.xaml.cs 46](#_Toc106488670)

[PrimeiraPagina.xaml.cs 49](#_Toc106488671)

[5. Bibliografia 52](#_Toc106488672)

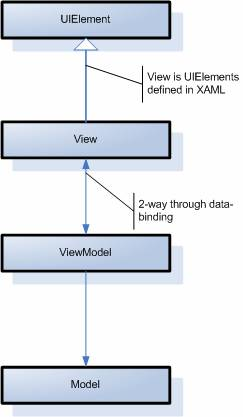
# Introdução

No fim deste projeto, após ter desenvolvido os Modelos (Models) e ter desenhado e construído as Views, faltava apenas o último passo lógico deste projeto, que era criar a ligação entre estes dois, utilizando assim, o componente ViewModel, pertencente à arquitetura MVVM, que estamos a utilizar no desenvolvimento deste projeto. Para tal, voltamos a utilizar a recomendação do docente da UC, e utilizar uma Framework para nos auxiliar, e ao mesmo tempo, nos preparar e enriquecer para o mundo do trabalho que nos espera.

Desta vez, a Framework a utilizar foi **ReactiveUI**, uma Framework declarativa de programação reativa (excelente para a arquitetura de software que nos foi requirida), muito utilizada no ecossistema .NET, da qual estamos a usar o WPF e o Xamarin.

# Enquadramento Teórico

Para entender o conceito principal desta etapa (ViewModel), temos de entender primeiro como esta se integra no conceito de MVVM. Aqui, a(s) View(s) (da segunda etapa), e o(s) Model(s) da primeira etapa, não se “conhecem”, ou seja, não interagem diretamente. Aqui, estes dois componentes estão unidos através do elusivo ViewModel, que conhece os dois, e destes, apenas a View conhece o ViewModel, e comunicam um com o outro através de um conceito conhecido por **bindings**.



Bindings são um mecanismo que se pode implementar em diversas arquiteturas de software, onde vários componentes gráficos se vão associar, de algum modo, a uma propriedade definida em um modelo, e assim, qualquer que seja a informação apresentada no mesmo ou inserida através deste, pelo utilizador, vai atualizar e representar os mesmos dados na respetiva propriedade no seu modelo. Deste modo, é possível simplificar a manipulação de dados e a sua representação, pois é possível utilizar diversas técnicas, frameworks e ferramentas para implementar este sistema.

É aqui que entra o **ReactiveUI**, pois embora o sistema “predefinido” de criar bindings com XAML + WPF, já seja extremamente simples e fácil de implementar, tem o aspeto negativo de criar uma forte (praticamente inseparável) associação entre estes dois componentes, ou seja, ao implementar de uma forma dita “normal” bindings no XAML, este ficaria muito dependente do seu Model ou ViewModel, sendo depois necessária quase uma reconstrução total da lógica caso fosse necessária alteração da mesma.

Com **ReactiveUI**, a ligação (ou binding), é feita apenas quando o ViewModel é ativado e detetado, e toda a interação é feita no mesmo, (necessitando apenas de ser referido no code-behind) de modo que não exista sequer uma linha no ficheiro da View (ficheiro XAML) que referencie qualquer tipo de função ou interação.

(Exemplo de código usando ReactiveUI)

[this](https://www.reactiveui.net/docs/handbook/view-models/).[WhenAnyValue](https://www.reactiveui.net/docs/handbook/when-any/)(x => x.SearchQuery)

.[Throttle](https://reactivex.io/documentation/operators/debounce.html)(TimeSpan.FromSeconds(0.8), RxApp.[TaskpoolScheduler](https://www.reactiveui.net/docs/handbook/scheduling/))

.[Select](https://reactivex.io/documentation/operators/map.html)(query => query?.Trim())

.[DistinctUntilChanged](https://reactivex.io/documentation/operators/distinct.html)()

.[Where](https://reactivex.io/documentation/operators/filter.html)(query => !string.IsNullOrWhiteSpace(query))

.[ObserveOn](https://reactivex.io/documentation/operators/observeon.html)(RxApp.[MainThreadScheduler](https://www.reactiveui.net/docs/handbook/scheduling/))

.[InvokeCommand](https://www.reactiveui.net/docs/handbook/commands/)(ExecuteSearch);

Para guardar e manipular dados, inicialmente iriamos utilizar uma base de dados, e assim, criar uma total sincronização entre as duas plataformas neste projeto, mas devido à ambiciosidade do mesmo, esse requisito foi deixado para trás, ficando apenas a necessidade de criar uma base de dados local, utilizando ficheiros XML, e para tal, utilizar a tecnologia **LINQ** que pode ser implementada na plataforma .NET

Em **LINQ**, podemos utilizar “queries” semelhantes às de linguagens de manipulação de dados (utilizados em várias linguagens de sistemas de gestão e manipulação de dados, como **SQL**).

Deste modo, conseguimos aceder, armazenar e manipular dados locais e online de uma forma simples eficiente e reutilizável.

# Objetivos da Etapa do Trabalho Prático

Como já referido, foi nos pedida a construção do ViewModel, e assim, fazer a ligação entre os Models e Views criados nas etapas anteriores. Seguindo o nosso método de trabalho anterior, para adquirir o máximo de conhecimento e capacidades possível, continuamos a seguir sempre as recomendações do docente da UC, e utilizamos todas as tecnologias recomendadas pelo mesmo, pois foi referido no início da etapa que estas recomendações não apenas tornam o processo de desenvolvimento mais suave, mas fazem com que estejamos no futuro, mais preparados para o mundo de o trabalho na área do desenvolvimento de software.

Embora opcional, um dos requisitos era a utilização da Framework **ReactiveUI**, que facilita a implementação de ViewModels no modelo MVVM na criação de aplicações no ecossistema .NET, neste caso, Xamarin e WPF, e foi isso mesmo que utilizamos.

Nos requisitos não funcionais, agora, as mockups que realizamos na etapa anterior, iriam ter de se transformar em componentes gráficas utilizáveis em aplicações funcionais, que representariam os dados definidos nos modelos também criados por nós. As funcionalidades principais eram a criação e respetiva autenticação de utilizadores, caso o utilizador mais tarde pretendesse, teria de existir a opção de editar essa conta, tal como a de a eliminar completamente, de um sistema local de dados no formato XML. Usando esta mesma tecnologia de armazenamento de dados, a função principal da aplicação iria se relacionar com listas de produtos, onde as principais funcionalidades seriam, a criação de listas, tal como a possibilidade de as editar e eliminar, e depois, a criação de produtos (tal como a edição e eliminação dos mesmos), e a sua associação às respetivas listas. Nos produtos, deveria ser possível associar uma categoria para mais fácil catalogação e organização/ordenação, e deste modo um conjunto de categorias predefinidas deferia estar disponível, tal como a opção de deixar o utilizador usar uma categoria personalizável, ou seja, criada pelo próprio.

Estas funcionalidades deviam estar disponíveis tanto na aplicação Desktop WPF, como na móvel Xamarin, onde não só os Models seriam os mesmos, mas o código do ViewModel deveria funcionar para ambas plataformas também, mas devido à nossa ainda pouca afinidade com a Framework, apenas conseguimos implementar todas as funcionalidades em WPF, não criando assim esta total separação lógica/gráfica que desejávamos, e sabemos ser não só possível, mas tecnologicamente fascinante.

# Desenvolvimento

## WPF

### ViewModel.cs

using ReactiveUI;

using ReactiveUI.Validation.Abstractions;

using ReactiveUI.Validation.Contexts;

using ReactiveUI.Validation.Extensions;

using System;

using System.Collections.Generic;

using System.Reactive;

using System.Text;

using System.IO;

using System.Windows.Input;

using Utad.Lab.G04.Domain.Models;

using System.Xml;

using System.Xml.Linq;

using System.Linq;

namespace Utad.Lab.G04.Domain.ViewModels

{

public class ViewModel : ReactiveObject, IValidatableViewModel

{

public ReactiveCommand<Unit, bool> CmdLogin { get; }

public ReactiveCommand<Unit, bool> CmdLogout { get; }

public ReactiveCommand<Unit, bool> CmdRegister { get; }

public ReactiveCommand<Unit, bool> CmdEdit { get; set; }

public ReactiveCommand<Unit, bool> CmdEliminar { get; set; }

public ReactiveCommand<Unit, bool> CmdCloseDlg { get; }

public ReactiveCommand<Unit, bool> CmdOpenRegisto { get; }

public ReactiveCommand<Unit, bool> CmdOpenEditarPerfil { get; }

public ReactiveCommand<Unit, bool> CmdOpenAddLista { get; }

public ReactiveCommand<Unit, bool> CmdOpenAddItem { get; set; }

public ReactiveCommand<Unit, bool> CmdShowConta { get; }

public ReactiveCommand<Unit, bool> CmdShowCats { get; }

public ReactiveCommand<Unit, bool> CmdShowListasDash { get; }

public ReactiveCommand<Unit, bool> CmdIsLogged { get; }

public ReactiveCommand<Unit, string> CmdAddFoto { get; set; }

public ReactiveCommand<Unit, bool> CmdAddLista { get; set; }

public ReactiveCommand<Unit, bool> CmdEditLista { get; set; }

public ReactiveCommand<Unit, bool> CmdDelLista { get; set; }

public ReactiveCommand<Unit, bool> CmdAddItem { get; set; }

public ReactiveCommand<Unit, bool> CmdEditItem { get; set; }

public ReactiveCommand<Unit, bool> CmdDelItem { get; set; }

public ReactiveCommand<Unit, bool> CmdMostraItems { get; }

public ReactiveCommand<Unit, bool> CmdMostraListas { get; set; }

public ValidationContext ValidationContext { get; } = new ValidationContext();

private User Utilizador;

public Categorias Cat;

public Dictionary<string, Lista> Listas { get; private set; }

public Dictionary<string, Produto> Produtos { get; private set; }

public ViewModel()

{

Utilizador = new User();

Cat = new Categorias();

Listas = new Dictionary<string, Lista>();

Produtos = new Dictionary<string, Produto>();

// Creates the validation for the Name property.

//this.ValidationRule(

// vm => vm.Email,

// ErroCredencias => !string.IsNullOrWhiteSpace(ErroCredencias),

// "Email Inválido");

//this.ValidationRule(

// vm => vm.Password,

// ErroCredencias => !string.IsNullOrWhiteSpace(ErroCredencias),

// "Password Incorreta");

CmdLogin = ReactiveCommand.Create(Login);

CmdLogout = ReactiveCommand.Create(Logout);

CmdRegister = ReactiveCommand.Create(Register);

CmdEliminar = ReactiveCommand.Create(Eliminar);

CmdEdit = ReactiveCommand.Create(Edit);

CmdCloseDlg = ReactiveCommand.Create(CloseDialogWindow);

CmdOpenRegisto = ReactiveCommand.Create(ShowDialogWindow);

CmdOpenEditarPerfil = ReactiveCommand.Create(ShowDialogWindow);

CmdOpenAddItem = ReactiveCommand.Create(ShowDialogWindow);

CmdAddItem = ReactiveCommand.Create(AddItem);

CmdEditItem = ReactiveCommand.Create(EditItem);

CmdDelItem = ReactiveCommand.Create(DelItem);

CmdOpenAddLista = ReactiveCommand.Create(ShowDialogWindow);

CmdShowConta = ReactiveCommand.Create(ShowDialogWindow);

CmdShowCats = ReactiveCommand.Create(MostraCats);

CmdShowListasDash = ReactiveCommand.Create(ShowDialogWindow);

CmdIsLogged = ReactiveCommand.Create(isLogged);

CmdAddFoto = ReactiveCommand.Create(AddFoto);

CmdAddLista = ReactiveCommand.Create(AddLista);

CmdEditLista = ReactiveCommand.Create(EditLista);

CmdDelLista = ReactiveCommand.Create(DelLista);

CmdMostraItems = ReactiveCommand.Create(MostraItems);

CmdMostraListas = ReactiveCommand.Create(MostraListas);

}

private string nome;

public string Nome

{

get => nome;

set => this.RaiseAndSetIfChanged(ref nome, value);

}

private string pais;

public string Pais

{

get => pais;

set => this.RaiseAndSetIfChanged(ref pais, value);

}

private string email;

public string Email

{

get => email;

set => this.RaiseAndSetIfChanged(ref email, value);

}

private string password;

public string Password

{

get => password;

set => this.RaiseAndSetIfChanged(ref password, value);

}

private string passwordconfirm;

public string PasswordConfirm

{

get => passwordconfirm;

set => this.RaiseAndSetIfChanged(ref passwordconfirm, value);

}

private string pfp;

public string PFP

{

get => pfp;

set => this.RaiseAndSetIfChanged(ref pfp, value);

}

private string erro\_credenciais;

public string ErroCredenciais

{

get => erro\_credenciais;

set => this.RaiseAndSetIfChanged(ref erro\_credenciais, value);

}

private bool is\_logged;

public bool IsLogged

{

get => is\_logged;

set => this.RaiseAndSetIfChanged(ref is\_logged, value);

}

//Listas

private string listadescricao;

public string ListaDescricao

{

get => listadescricao;

set => this.RaiseAndSetIfChanged(ref listadescricao, value);

}

private string ld;

public string LD

{

get => ld;

set => this.RaiseAndSetIfChanged(ref ld, value);

}

//Produtos / Items

private string itemdescricao;

public string ItemDescricao

{

get => itemdescricao;

set => this.RaiseAndSetIfChanged(ref itemdescricao, value);

}

private string pd;

public string PD

{

get => pd;

set => this.RaiseAndSetIfChanged(ref pd, value);

}

private int itemquantidade;

public int ItemQuantidade

{

get => itemquantidade;

set => this.RaiseAndSetIfChanged(ref itemquantidade, value);

}

private string pq;

public string PQ

{

get => pq;

set => this.RaiseAndSetIfChanged(ref pq, value);

}

private string itemcat;

public string ItemCat

{

get => itemcat;

set => this.RaiseAndSetIfChanged(ref itemcat, value);

}

private string pc;

public string PC

{

get => pc;

set => this.RaiseAndSetIfChanged(ref pc, value);

}

private bool pcomprado;

public bool PComprado

{

get => pcomprado;

set => this.RaiseAndSetIfChanged(ref pcomprado, value);

}

//Implementação de Comandos

//---------------------------------------------------

private bool isLogged()

{

if (File.Exists("users.xml"))

{

XDocument doc = XDocument.Load("users.xml");

var registo = from user in doc.Element("Users").Descendants("User") select user;

foreach (var campos in registo)

{

if (campos.Element("IsLogged") != null && campos.Element("IsLogged").Value == "true")

{

Nome = campos.Element("Nome").Value;

Email = campos.Element("Email").Value;

IsLogged = true;

return true;

}

}

return false;

}

return false;

}

private bool Login()

{

if (File.Exists("users.xml"))

{

XDocument doc = XDocument.Load("users.xml");

var registo = from user in doc.Element("Users").Descendants("User") where user.Element("Email").Value == Email select user;

foreach (var campos in registo)

{

if (campos.Element("Email") != null && campos.Element("Password") != null && (string)campos.Element("Email") == Email && (string)campos.Element("Password") == Password && !string.IsNullOrEmpty(Email) && !string.IsNullOrEmpty(Password))

{

campos.SetElementValue("IsLogged", "true");

doc.Save("users.xml");

ErroCredenciais = " ";

IsLogged = true;

isLogged();

return true;

}

else

{

Email = "";

Password = "";

ErroCredenciais = "Credenciais Erradas";

return false;

}

}

Email = "";

Password = "";

ErroCredenciais = "Credenciais Erradas";

return false;

}

else

{

Email = "";

Password = "";

ErroCredenciais = "Credenciais Erradas";

return false;

}

}

private bool Logout()

{

if (File.Exists("users.xml"))

{

XDocument doc = XDocument.Load("users.xml");

var registo = from user in doc.Element("Users").Descendants("User") select user;

foreach (var campos in registo)

{

campos.SetElementValue("IsLogged", "false");

doc.Save("users.xml");

IsLogged = false;

return true;

};

return false;

}

else

{

return false;

}

}

private bool Edit()

{

if (!string.IsNullOrEmpty(Email) && !string.IsNullOrEmpty(Nome) && !string.IsNullOrEmpty(Password) && !string.IsNullOrEmpty(PasswordConfirm))

{

if (Password == PasswordConfirm)

{

ErroCredenciais = " ";

Utilizador.Nome = Nome;

Utilizador.Email = Email;

Utilizador.Password = Password;

XDocument doc = XDocument.Load("users.xml");

var registo = from user in doc.Element("Users").Descendants("User") where user.Element("IsLogged").Value == "true" select user;

foreach (var campos in registo)

{

campos.SetElementValue("Nome", Nome);

campos.SetAttributeValue("Email", Email);

campos.SetElementValue("Email", Email);

campos.SetElementValue("Password", Password);

}

doc.Save("users.xml");

return true;

}

else

{

Password = "";

PasswordConfirm = "";

ErroCredenciais = "Passwords não coincidem";

return false;

}

}

else

{

Email = "";

Password = "";

ErroCredenciais = "Credenciais Inválidas";

return false;

}

}

private bool Eliminar()

{

if (File.Exists("users.xml"))

{

XDocument doc = XDocument.Load("users.xml");

var registo = from user in doc.Element("Users").Descendants("User") where user.Element("IsLogged").Value == "true" select user;

foreach (var campos in registo)

{

campos.Remove();

doc.Save("users.xml");

return true;

}

}

else

return false;

return false;

}

private string AddFoto()

{

return "pfp";

}

private bool ShowDialogWindow()

{

return true;

}

private bool CloseDialogWindow()

{

return false;

}

private bool Register()

{

if (!string.IsNullOrEmpty(Email) && !string.IsNullOrEmpty(Nome) && !string.IsNullOrEmpty(Password) && !string.IsNullOrEmpty(PasswordConfirm))

{

if (Password == PasswordConfirm)

{

ErroCredenciais = " ";

Utilizador.Nome = Nome;

Utilizador.Pais = Pais;

Utilizador.Email = Email;

Utilizador.Password = Password;

if (!File.Exists("users.xml"))

{

XDocument doc = new XDocument(

new XElement("Users",

new XElement("User",

new XAttribute("Email", Email),

new XElement("Nome", Nome),

new XElement("Pais", Pais),

new XElement("Email", Email),

new XElement("Password", Password),

new XElement("IsLogged", "false")

)

)

);

doc.Save("users.xml");

}

else

{

XDocument doc = XDocument.Load("users.xml");

XElement XnovoUser;

XnovoUser = new XElement("User",

new XAttribute("Email", Email),

new XElement("Nome", Nome),

new XElement("Pais", Pais),

new XElement("Email", Email),

new XElement("Password", Password),

new XElement("IsLogged", "false")

);

doc.Element("Users").Add(XnovoUser);

doc.Save("users.xml");

return true;

}

return true;

}

else

{

Password = "";

PasswordConfirm = "";

ErroCredenciais = "Passwords não coincidem";

return false;

}

}

else

{

Email = "";

Password = "";

ErroCredenciais = "Credenciais Inválidas";

return false;

}

}

private bool AddLista()

{

if (!string.IsNullOrEmpty(ListaDescricao) || !string.IsNullOrEmpty(LD))

{

juntaListas();

return true;

}

return false;

}

private bool EditLista()

{

XDocument doc = XDocument.Load("listas.xml");

var listas = from lista in doc.Element("Listas").Descendants("Lista")

where lista.Attribute("Descricao").Value == LD

select lista;

foreach (var l in listas)

{

l.Attribute("Descricao").Value = ListaDescricao;

doc.Save("listas.xml");

return true;

}

return false;

}

private bool DelLista()

{

XDocument doc = XDocument.Load("listas.xml");

var listas = from lista in doc.Element("Listas").Descendants("Lista")

where lista.Attribute("Descricao").Value == LD

select lista;

foreach (var l in listas)

{

l.Remove();

doc.Save("listas.xml");

return true;

}

return false;

}

private bool DelItem()

{

XDocument doc = XDocument.Load("listas.xml");

var listas = from lista in doc.Element("Listas").Descendants("Lista")

where lista.Attribute("Descricao").Value == LD

select lista;

foreach (var l in listas)

{

var items = from item in l.Descendants("Item")

where item.Element("Descricao").Value == PD

select item;

foreach (var i in items)

{

i.Remove();

doc.Save("listas.xml");

return true;

}

}

return false;

}

private bool EditItem()

{

XDocument doc = XDocument.Load("listas.xml");

var listas = from lista in doc.Element("Listas").Descendants("Lista")

where lista.Attribute("Descricao").Value == ListaDescricao

select lista;

foreach (var l in listas)

{

var items = from item in l.Descendants("Item")

where item.Element("Descricao").Value == ItemDescricao

select item;

foreach (var i in items)

{

i.Element("Descricao").Value = PD;

i.Element("Quantidade").Value = PQ;

i.Element("Categoria").Value = PC;

doc.Save("listas.xml");

return true;

}

}

return false;

}

private bool AddItem()

{

if (!string.IsNullOrEmpty(ItemDescricao) && ItemQuantidade > 0 && !string.IsNullOrEmpty(ItemCat) && !string.IsNullOrEmpty(ListaDescricao))

{

if (File.Exists("listas.xml"))

{

XDocument doc = XDocument.Load("listas.xml");

juntaListas();

XElement XnovoItem;

XnovoItem = new XElement("Item",

new XElement("Descricao", ItemDescricao),

new XElement("Quantidade", ItemQuantidade),

new XElement("Categoria", ItemCat)

);

var listas = from lista in doc.Element("Listas").Descendants("Lista")

where lista.Attribute("Descricao").Value == ListaDescricao

select lista;

foreach (var l in listas)

{

l.Add(XnovoItem);

doc.Save("listas.xml");

return true;

}

//se existir o ficheiro, mas a lista não existir

XElement XnovaLista;

XnovaLista = new XElement("Lista",

new XAttribute("Descricao", ListaDescricao)

);

doc.Element("Listas").Add(XnovaLista);

doc.Save("listas.xml");

juntaListas();

listas = from lista in doc.Element("Listas").Descendants("Lista")

where lista.Attribute("Descricao").Value == ListaDescricao

select lista;

foreach (var l in listas)

{

l.Add(XnovoItem);

doc.Save("listas.xml");

return true;

}

}

else

{

XDocument doc = new XDocument(

new XElement("Listas",

new XElement("Lista",

new XAttribute("Descricao", ListaDescricao)

)

)

);

doc.Save("listas.xml");

juntaListas();

XElement XnovoItem;

XnovoItem = new XElement("Item",

new XElement("Descricao", ItemDescricao),

new XElement("Quantidade", ItemQuantidade),

new XElement("Categoria", ItemCat)

);

var listas = from lista in doc.Element("Listas").Descendants("Lista")

where lista.Attribute("Descricao").Value == ListaDescricao

select lista;

foreach (var l in listas)

{

l.Add(XnovoItem);

doc.Save("listas.xml");

return true;

}

}

}

return false;

}

private bool MostraListas()

{

if (File.Exists("listas.xml"))

{

juntaListas();

return true;

}

else

{

return false;

}

}

private bool MostraItems()

{

if (File.Exists("listas.xml"))

{

juntaItems();

return true;

}

else

{

return false;

}

}

private bool juntaListas()

{

XDocument doc = XDocument.Load("listas.xml");

var registo = from lista in doc.Element("Listas").Descendants("Lista")

select lista;

foreach (var campos in registo)

{

if (!Listas.ContainsKey(campos.Attribute("Descricao").Value))

{

Lista novaLista = new Lista();

novaLista.Descricao = campos.Attribute("Descricao").Value;

Listas.Add(novaLista.Descricao, novaLista);

}

};

return true;

}

private bool juntaItems()

{

XDocument doc = XDocument.Load("listas.xml");

var listas = from lista in doc.Element("Listas").Descendants("Lista")

where lista.Attribute("Descricao").Value == ListaDescricao

select lista;

foreach (var l in listas)

{

var items = from i in l.Descendants("Item")

select i;

foreach (var i in items)

{

if (!Produtos.ContainsKey(i.Element("Descricao").Value))

{

Produto novoItem = new Produto();

novoItem.Descricao = i.Element("Descricao").Value;

novoItem.Quantidade = Convert.ToInt32(i.Element("Quantidade").Value);

novoItem.Categoria = i.Element("Categoria").Value;

Produtos.Add(novoItem.Descricao, novoItem);

}

// return true;

}

}

return false;

}

private bool MostraCats()

{

return true;

}

}

}

### Login.xaml.cs

using ReactiveUI;

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 Syncfusion.Windows.Shared;

using Utad.Lab.G04.Domain.ViewModels;

using System.Reactive.Disposables;

using System.Reactive.Linq;

namespace Utad.Lab.G04.DesktopWPF

{

public partial class Login : ChromelessWindow, IViewFor<ViewModel>

{

public static readonly DependencyProperty ViewModelProperty = DependencyProperty

.Register(nameof(ViewModel), typeof(ViewModel), typeof(Login));

public ViewModel ViewModel

{

get => (ViewModel)GetValue(ViewModelProperty);

set => SetValue(ViewModelProperty, value);

}

object IViewFor.ViewModel

{

get => ViewModel;

set => ViewModel = (ViewModel)value;

}

public Login()

{

InitializeComponent();

ViewModel = new ViewModel();

this.WhenActivated(disposable => //disposable para evitar memory leaks

{

//vm: ViewModel, v: View (Login)

//Bind do Email

this.Bind(ViewModel, vm => vm.Email, v => v.TBMail.Text).DisposeWith(disposable);

//Bind da Password

this.Bind(ViewModel, vm => vm.Password, v => v.TBPassword.Text).DisposeWith(disposable);

//Bind do Erro: Apenas usa One Way

this.OneWayBind(ViewModel, vm => vm.ErroCredenciais, v => v.LabelLoginFailed.Content).DisposeWith(disposable);

//Bind do Comando para Login

this.BindCommand(ViewModel, vm => vm.CmdLogin, v => v.BTNLogin).DisposeWith(disposable);

//Bind do Comando para Registo

this.BindCommand(ViewModel, vm => vm.CmdOpenRegisto, v => v.BTNRegister).DisposeWith(disposable);

});

//verifica as credenciais

ViewModel.CmdLogin

.Where(val => val != false)

.Subscribe(val =>

{

DialogResult = val;

});

//abre o registo

ViewModel.CmdOpenRegisto

.Where(val => val != false)

.Subscribe(val => { ViewModel.ErroCredenciais = ""; this.Hide(); Registar dlg = new Registar(); if (dlg.ShowDialog() == true) { this.ShowDialog(); } });

}

}

}

### Registar.xaml.cs

using System;

using System.Collections.Generic;

using System.IO;

using System.Linq;

using System.Reactive.Disposables;

using System.Reactive.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 Microsoft.Win32;

using ReactiveUI;

using Syncfusion.Windows.Shared;

using Utad.Lab.G04.Domain.ViewModels;

namespace Utad.Lab.G04.DesktopWPF

{

public partial class Registar : ChromelessWindow, IViewFor<ViewModel>

{

public static readonly DependencyProperty ViewModelProperty = DependencyProperty

.Register(nameof(ViewModel), typeof(ViewModel), typeof(Registar));

public ViewModel ViewModel

{

get => (ViewModel)GetValue(ViewModelProperty);

set => SetValue(ViewModelProperty, value);

}

object IViewFor.ViewModel

{

get => ViewModel;

set => ViewModel = (ViewModel)value;

}

public Registar()

{

InitializeComponent();

ViewModel = new ViewModel();

this.WhenActivated(disposable => //disposable para evitar memory leaks

{

//vm: ViewModel, v: View (Register)

//Bind do Email

this.Bind(ViewModel, vm => vm.Email, v => v.TBMail.Text).DisposeWith(disposable);

//Bind do Nome

this.Bind(ViewModel, vm => vm.Nome, v => v.TBNome.Text).DisposeWith(disposable);

//Bind do Pa�s

this.Bind(ViewModel, vm => vm.Pais, v => v.TBPais.Text).DisposeWith(disposable);

//Bind da Password

this.Bind(ViewModel, vm => vm.Password, v => v.TBPassword.Text).DisposeWith(disposable);

//Bind da Password Confirm

this.Bind(ViewModel, vm => vm.PasswordConfirm, v => v.TBPasswordConf.Text).DisposeWith(disposable);

//Bind da Password Confirm

this.OneWayBind(ViewModel, vm => vm.ErroCredenciais, v => v.LabelPasswordError.Content).DisposeWith(disposable);

//Bind do Comando para Foto

this.BindCommand(ViewModel, vm => vm.CmdAddFoto, v => v.BTNProfilePic).DisposeWith(disposable);

//Bind do Comando para Registo

this.BindCommand(ViewModel, vm => vm.CmdRegister, v => v.BTNRegister).DisposeWith(disposable);

});

//abre o registo

ViewModel.CmdRegister

.Where(val => val != false)

.Subscribe(val => DialogResult = val);

ViewModel.CmdAddFoto

.Where(file => !string.IsNullOrEmpty(file))

.Subscribe(file =>

{

OpenFileDialog fotoPerfil = new OpenFileDialog();

fotoPerfil.Filter = "Imagens|\*.jpg;\*.jpeg;\*.png;...";

if (fotoPerfil.ShowDialog() == true)

{

IMGperfil.ImageSource = new BitmapImage(new Uri(fotoPerfil.FileName));

File.Copy(fotoPerfil.FileName, "pfp" + System.IO.Path.GetExtension(fotoPerfil.FileName));

}

});

}

}

}

### EditarConta.xaml.cs

using System;

using System.Collections.Generic;

using System.Linq;

using System.Reactive.Disposables;

using System.Reactive.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 Microsoft.Win32;

using ReactiveUI;

using Syncfusion.Windows.Shared;

using Utad.Lab.G04.Domain.ViewModels;

namespace Utad.Lab.G04.DesktopWPF

{

/// <summary>

/// Interaction logic for EditarConta.xaml

/// </summary>

///

public partial class EditarConta : ChromelessWindow, IViewFor<ViewModel>

{

public static readonly DependencyProperty ViewModelProperty = DependencyProperty

.Register(nameof(ViewModel), typeof(ViewModel), typeof(EditarConta));

public ViewModel ViewModel

{

get => (ViewModel)GetValue(ViewModelProperty);

set => SetValue(ViewModelProperty, value);

}

object IViewFor.ViewModel

{

get => ViewModel;

set => ViewModel = (ViewModel)value;

}

public EditarConta(string \_Nome, string \_Email)

{

InitializeComponent();

ViewModel = new ViewModel();

this.WhenActivated(disposable => //disposable para evitar memory leaks

{

//vm: ViewModel, v: View (Register)

//TBNome.Text = \_Nome;

//TBMail.Text = \_Email;

ViewModel.Nome = \_Nome;

ViewModel.Email = \_Email;

//Bind do Email

this.Bind(ViewModel, vm => vm.Email, v => v.TBMail.Text).DisposeWith(disposable);

//Bind do Nome

this.Bind(ViewModel, vm => vm.Nome, v => v.TBNome.Text).DisposeWith(disposable);

//Bind da Password

this.Bind(ViewModel, vm => vm.Password, v => v.TBPassword.Text).DisposeWith(disposable);

//Bind da Password Confirm

this.Bind(ViewModel, vm => vm.PasswordConfirm, v => v.TBPasswordConf.Text).DisposeWith(disposable);

//Bind da Password Confirm

this.OneWayBind(ViewModel, vm => vm.ErroCredenciais, v => v.LabelPasswordError.Content).DisposeWith(disposable);

//Bind do Comando para Alterar

this.BindCommand(ViewModel, vm => vm.CmdEdit, v => v.BTNEditar).DisposeWith(disposable);

});

//abre o edit

ViewModel.CmdEdit

.Where(val => val != false)

.Subscribe(val =>

{

DialogResult = val;

});

}

}

}

### CriarLista.xaml.cs

using System;

using System.Collections;

using System.Collections.Generic;

using System.Linq;

using System.Reactive.Disposables;

using System.Reactive.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 ReactiveUI;

using Syncfusion.SfSkinManager;

using Syncfusion.Windows.Shared;

using Syncfusion.Windows.Tools.Controls;

using Utad.Lab.G04.Domain.ViewModels;

namespace Utad.Lab.G04.DesktopWPF

{

public partial class CriarLista : ChromelessWindow, IViewFor<ViewModel>

{

public static readonly DependencyProperty ViewModelProperty = DependencyProperty

.Register(nameof(ViewModel), typeof(ViewModel), typeof(CriarLista));

public ViewModel ViewModel

{

get => (ViewModel)GetValue(ViewModelProperty);

set => SetValue(ViewModelProperty, value);

}

object IViewFor.ViewModel

{

get => ViewModel;

set => ViewModel = (ViewModel)value;

}

public CriarLista(string titulo)

{

InitializeComponent();

SfSkinManager.SetTheme(this, new FluentTheme() { ThemeName = "FluentDark", ShowAcrylicBackground = true });

ViewModel = new ViewModel();

this.WhenActivated(disposable => //disposable para evitar memory leaks

{

//vm: ViewModel, v: View (CriarLista)

this.Bind(ViewModel, vm => vm.ListaDescricao, v => v.TBDesc.Text).DisposeWith(disposable);

//Bind do Comando para Adicionar Produto

this.BindCommand(ViewModel, vm => vm.CmdOpenAddItem, v => v.BTNAddItem).DisposeWith(disposable);

this.BindCommand(ViewModel, vm => vm.CmdCloseDlg, v => v.BTNCancela).DisposeWith(disposable);

if (!string.IsNullOrEmpty(titulo))

{

ViewModel.LD = titulo;

TBDesc.Text = titulo;

NL.Content = "Editar Lista";

CVItems.Items.Clear();

ViewModel.Produtos.Clear();

ViewModel.CmdMostraItems.Execute().Subscribe();

foreach (var item in ViewModel.Produtos)

{

Button delBTN = new Button();

delBTN.Content = "Apagar";

delBTN.Width = 50;

delBTN.Margin = new Thickness(2);

delBTN.Name = item.Value.Descricao + "\_" +

item.Value.Quantidade.ToString() + "\_" +

item.Value.Categoria.Replace(" ", "\_");

delBTN.HorizontalAlignment = HorizontalAlignment.Right;

delBTN.Click += DelBTN\_Click;

Button editBTN = new Button();

editBTN.Content = "Editar";

editBTN.Width = 50;

editBTN.Margin = new Thickness(2);

editBTN.Name = item.Value.Descricao + "\_" +

item.Value.Quantidade.ToString() + "\_" +

item.Value.Categoria.Replace(" ", "\_"); ;

editBTN.HorizontalAlignment = HorizontalAlignment.Right;

editBTN.Click += EditBTN\_Click;

CardViewItem CVI = new CardViewItem();

CVI.Header = item.Key;

CVI.HorizontalAlignment = HorizontalAlignment.Right;

Label desc = new Label();

desc.Content = "Quantidade: " + item.Value.Quantidade.ToString()

+ "\n"

+ "Categoria: " + item.Value.Categoria.ToString();

StackPanel SP = new StackPanel();

SP.Children.Add(desc);

SP.Children.Add(editBTN);

SP.Children.Add(delBTN);

CVI.Content = SP;

CVItems.Items.Add(CVI);

}

}

switch (NL.Content)

{

case "Nova Lista":

//Bind das Listas (adicionar)

//Bind do Comando para Adicionar Lista

//Bind da Descricao

this.Bind(ViewModel, vm => vm.ListaDescricao, v => v.TBDesc.Text).DisposeWith(disposable);

this.BindCommand(ViewModel, vm => vm.CmdAddLista, v => v.BTNConcluido).DisposeWith(disposable);

break;

case "Editar Lista":

//Bind das Listas (para editar)

//Bind da Descricao

this.BindCommand(ViewModel, vm => vm.CmdEditLista, v => v.BTNConcluido).DisposeWith(disposable);

break;

}

});

ViewModel.CmdAddLista

.Where(val => val == true)

.Subscribe(val =>

{

ErrorLBL.Visibility = Visibility.Hidden;

DialogResult = val;

this.Close();

});

ViewModel.CmdAddLista

.Where(val => val == false)

.Subscribe(val =>

{

ErrorLBL.Visibility = Visibility.Visible;

});

ViewModel.CmdEditLista

.Where(val => val == true)

.Subscribe(val =>

{

ErrorLBL.Visibility = Visibility.Hidden;

DialogResult = val;

this.Close();

});

ViewModel.CmdEditLista

.Where(val => val == false)

.Subscribe(val =>

{

ErrorLBL.Visibility = Visibility.Visible;

});

ViewModel.CmdCloseDlg

.Where(val => val != true)

.Subscribe(val => { this.Close(); });

ViewModel.CmdOpenAddItem

.Where(val => val == true)

.Subscribe(val =>

{

JanelaProduto dlg = new JanelaProduto(TBDesc.Text, "", "", ViewModel.ListaDescricao); if (dlg.ShowDialog() == val)

{

this.Show();

CVItems.Items.Clear();

ViewModel.Produtos.Clear();

ViewModel.CmdMostraItems.Execute().Subscribe();

foreach (var item in ViewModel.Produtos)

{

Button delBTN = new Button();

delBTN.Content = "Apagar";

delBTN.Width = 50;

delBTN.Margin = new Thickness(2);

delBTN.Name = item.Value.Descricao + "\_" +

item.Value.Quantidade.ToString() + "\_" +

item.Value.Categoria.Replace(" ", "\_");

delBTN.HorizontalAlignment = HorizontalAlignment.Right;

delBTN.Click += DelBTN\_Click;

Button editBTN = new Button();

editBTN.Content = "Editar";

editBTN.Width = 50;

editBTN.Margin = new Thickness(2);

editBTN.Name = item.Value.Descricao + "\_" +

item.Value.Quantidade.ToString() + "\_" +

item.Value.Categoria.Replace(" ", "\_"); ;

editBTN.HorizontalAlignment = HorizontalAlignment.Right;

editBTN.Click += EditBTN\_Click;

CardViewItem CVI = new CardViewItem();

CVI.Header = item.Key;

CVI.HorizontalAlignment = HorizontalAlignment.Right;

Label desc = new Label();

desc.Content = "Quantidade: " + item.Value.Quantidade.ToString()

+ "\n"

+ "Categoria: " + item.Value.Categoria.ToString();

StackPanel SP = new StackPanel();

SP.Children.Add(desc);

SP.Children.Add(editBTN);

SP.Children.Add(delBTN);

CVI.Content = SP;

CVItems.Items.Add(CVI);

}

}

});

}

private void DelBTN\_Click(object sender, RoutedEventArgs e)

{

Button button = (Button)sender;

var fullname = button.Name.Split("\_");

string name = fullname[0];

ViewModel.PD = name;

ViewModel.LD = ViewModel.ListaDescricao;

if (MessageBox.Show("Deseja apagar o item " + name + "?",

"Apagar Item",

MessageBoxButton.YesNo,

MessageBoxImage.Question) == MessageBoxResult.Yes)

{

ViewModel.CmdDelItem

.Execute()

.Where(val => val == true)

.Subscribe(val =>

{

MessageBox.Show("Item " + name + " apagado", "Item Apagado", MessageBoxButton.OK, MessageBoxImage.Information);

CVItems.Items.Clear();

ViewModel.Produtos.Clear();

ViewModel.CmdMostraItems.Execute().Subscribe();

foreach (var item in ViewModel.Produtos)

{

Button delBTN = new Button();

delBTN.Content = "Apagar";

delBTN.Width = 50;

delBTN.Margin = new Thickness(2);

delBTN.Name = item.Value.Descricao + "\_" +

item.Value.Quantidade.ToString() + "\_" +

item.Value.Categoria.Replace(" ", "\_");

delBTN.HorizontalAlignment = HorizontalAlignment.Right;

delBTN.Click += DelBTN\_Click;

Button editBTN = new Button();

editBTN.Content = "Editar";

editBTN.Width = 50;

editBTN.Margin = new Thickness(2);

editBTN.Name = item.Value.Descricao + "\_" +

item.Value.Quantidade.ToString() + "\_" +

item.Value.Categoria.Replace(" ", "\_"); ;

editBTN.HorizontalAlignment = HorizontalAlignment.Right;

editBTN.Click += EditBTN\_Click;

CardViewItem CVI = new CardViewItem();

CVI.Header = item.Key;

CVI.HorizontalAlignment = HorizontalAlignment.Right;

Label desc = new Label();

desc.Content = "Quantidade: " + item.Value.Quantidade.ToString()

+ "\n"

+ "Categoria: " + item.Value.Categoria.ToString();

StackPanel SP = new StackPanel();

SP.Children.Add(desc);

SP.Children.Add(editBTN);

SP.Children.Add(delBTN);

CVI.Content = SP;

CVItems.Items.Add(CVI);

}

});

}

}

private void EditBTN\_Click(object sender, RoutedEventArgs e)

{

Button button = (Button)sender;

var fullname = button.Name.Split("\_");

string desc = fullname[0];

string quant = fullname[1];

string cat = fullname[2];

if (fullname.Count() == 5)

{

cat = fullname[2] + "\_" + fullname[3] + "\_" + fullname[4];

}

ViewModel.PD = desc;

ViewModel.PQ = quant;

ViewModel.PC = cat;

ViewModel.LD = ViewModel.ListaDescricao;

JanelaProduto dlg = new JanelaProduto(desc, quant, cat, ViewModel.ListaDescricao);

if (dlg.ShowDialog() == true)

{

MessageBox.Show("Produto " + desc + " alterado", "Lista Alterada", MessageBoxButton.OK, MessageBoxImage.Information);

CVItems.Items.Clear();

ViewModel.Produtos.Clear();

ViewModel.CmdMostraItems.Execute().Subscribe();

foreach (var item in ViewModel.Produtos)

{

Button delBTN = new Button();

delBTN.Content = "Apagar";

delBTN.Width = 50;

delBTN.Margin = new Thickness(2);

delBTN.Name = item.Value.Descricao + "\_" +

item.Value.Quantidade.ToString() + "\_" +

item.Value.Categoria.Replace(" ", "\_");

delBTN.HorizontalAlignment = HorizontalAlignment.Right;

delBTN.Click += DelBTN\_Click;

Button editBTN = new Button();

editBTN.Content = "Editar";

editBTN.Width = 50;

editBTN.Margin = new Thickness(2);

editBTN.Name = item.Value.Descricao + "\_" +

item.Value.Quantidade.ToString() + "\_" +

item.Value.Categoria.Replace(" ", "\_"); ;

editBTN.HorizontalAlignment = HorizontalAlignment.Right;

editBTN.Click += EditBTN\_Click;

CardViewItem CVI = new CardViewItem();

CVI.Header = item.Key;

CVI.HorizontalAlignment = HorizontalAlignment.Right;

Label descL = new Label();

descL.Content = "Quantidade: " + item.Value.Quantidade.ToString()

+ "\n"

+ "Categoria: " + item.Value.Categoria.ToString();

StackPanel SP = new StackPanel();

SP.Children.Add(descL);

SP.Children.Add(editBTN);

SP.Children.Add(delBTN);

CVI.Content = SP;

CVItems.Items.Add(CVI);

}

}

}

}

}

### JanelaProduto.xaml.cs

using System;

using System.Collections.Generic;

using System.Collections.ObjectModel;

using System.Linq;

using System.Reactive.Disposables;

using System.Reactive.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 ReactiveUI;

using Syncfusion.SfSkinManager;

using Syncfusion.Windows.Shared;

using Syncfusion.Windows.Tools.Controls;

using Utad.Lab.G04.Domain.Models;

using Utad.Lab.G04.Domain.ViewModels;

namespace Utad.Lab.G04.DesktopWPF

{

public partial class JanelaProduto : ChromelessWindow, IViewFor<ViewModel>

{

public static readonly DependencyProperty ViewModelProperty = DependencyProperty

.Register(nameof(ViewModel), typeof(ViewModel), typeof(JanelaProduto));

public ViewModel ViewModel

{

get => (ViewModel)GetValue(ViewModelProperty);

set => SetValue(ViewModelProperty, value);

}

object IViewFor.ViewModel

{

get => ViewModel;

set => ViewModel = (ViewModel)value;

}

public JanelaProduto(string desc, string quant, string cat, string listaD)

{

InitializeComponent();

ViewModel = new ViewModel();

this.WhenActivated(disposable => //disposable para evitar memory leaks

{

//vm: ViewModel, v: View (JanelaProduto)

CBCat.SelectionChanged += CBCatSelectionChanged;

this.BindCommand(ViewModel, vm => vm.CmdShowCats, v => v.CBCat).DisposeWith(disposable);

this.BindCommand(ViewModel, vm => vm.CmdCloseDlg, v => v.BTNCancela).DisposeWith(disposable);

if (!string.IsNullOrEmpty(desc) && !string.IsNullOrEmpty(quant) && !string.IsNullOrEmpty(cat))

{

LBP.Content = "Editar Produto";

TBDesc.Text = desc;

ViewModel.PD = desc;

ViewModel.ItemDescricao = desc;

TBQuant.Text = quant;

ViewModel.PQ = quant;

CBCat.SelectedValue = cat.Replace("\_", " ");

}

ViewModel.PC = cat.Replace("\_", " ");

switch (LBP.Content)

{

case "Novo Produto":

//Bind dos Produtos (adicionar)

this.Bind(ViewModel, vm => vm.ItemDescricao, v => v.TBDesc.Text).DisposeWith(disposable);

this.Bind(ViewModel, vm => vm.ItemQuantidade, v => v.TBQuant.Text).DisposeWith(disposable);

if (NewCatBox.IsVisible)

{

this.Bind(ViewModel, vm => vm.ItemCat, v => v.TBNewCat.Text).DisposeWith(disposable);

CBCat.SelectedValue = "Outra";

}

else

{

this.Bind(ViewModel, vm => vm.ItemCat, v => v.CBCat.SelectedValue).DisposeWith(disposable);

}

this.BindCommand(ViewModel, vm => vm.CmdAddItem, v => v.BTNConcluido).DisposeWith(disposable);

break;

case "Editar Produto":

//Bind dos Produtos (para editar)

this.Bind(ViewModel, vm => vm.PD, v => v.TBDesc.Text).DisposeWith(disposable);

this.Bind(ViewModel, vm => vm.PQ, v => v.TBQuant.Text).DisposeWith(disposable);

if (NewCatBox.IsVisible)

{

this.Bind(ViewModel, vm => vm.PC, v => v.TBNewCat.Text).DisposeWith(disposable);

CBCat.SelectedValue = "Outra";

}

else

{

this.Bind(ViewModel, vm => vm.PC, v => v.CBCat.SelectedValue).DisposeWith(disposable);

}

this.BindCommand(ViewModel, vm => vm.CmdEditItem, v => v.BTNConcluido).DisposeWith(disposable);

break;

}

void CBCatSelectionChanged(object sender, SelectionChangedEventArgs e)

{

ComboBoxAdv cb = (ComboBoxAdv)sender;

if (CBCat.SelectedValue == null)

{

CBCat.SelectedValue = "Outra";

}

if (CBCat.SelectedValue.ToString() == "Outra")

{

NewCatBox.Visibility = Visibility.Visible;

TBNewCat.Visibility = Visibility.Visible;

this.Bind(ViewModel, vm => vm.PC, v => v.TBNewCat.Text).DisposeWith(disposable);

return;

}

else

{

NewCatBox.Visibility = Visibility.Hidden;

TBNewCat.Visibility = Visibility.Hidden;

CBCat.SelectedValue = cb.SelectedValue;

ViewModel.PC = cb.SelectedValue.ToString();

//this.Bind(ViewModel, vm => vm.PC, v => v.CBCat.SelectedValue).DisposeWith(disposable);

return;

}

}

});

ViewModel.ListaDescricao = listaD;

ViewModel.CmdAddItem

.Subscribe(val =>

{

if (val)

{

DialogResult = val;

this.Close();

}

else

{

LDadosErrados.Visibility = Visibility.Visible;

}

});

ViewModel.CmdEditItem

.Subscribe(val =>

{

if (val)

{

DialogResult = val;

this.Close();

}

else

{

LDadosErrados.Visibility = Visibility.Visible;

}

});

ViewModel.CmdCloseDlg

.Where(val => val != true)

.Subscribe(val => { this.Close(); });

ViewModel.CmdShowCats

.Execute()

.Where(val => val == true)

.Subscribe(val =>

{

foreach (var cat in ViewModel.Cat.Descricao)

{

CBCat.Items.Add(cat);

}

});

}

}

}

### MainWindow.xaml.cs

using ReactiveUI;

using Syncfusion.SfSkinManager;

using Syncfusion.Windows.Shared;

using Syncfusion.Windows.Tools.Controls;

using System;

using System.Collections.Generic;

using System.IO;

using System.Linq;

using System.Reactive.Disposables;

using System.Reactive.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 Utad.Lab.G04.Domain.ViewModels;

namespace Utad.Lab.G04.DesktopWPF

{

public partial class MainWindow : ChromelessWindow, IViewFor<ViewModel>

{

public static readonly DependencyProperty ViewModelProperty = DependencyProperty

.Register(nameof(ViewModel), typeof(ViewModel), typeof(MainWindow));

public ViewModel ViewModel

{

get => (ViewModel)GetValue(ViewModelProperty);

set => SetValue(ViewModelProperty, value);

}

object IViewFor.ViewModel

{

get => ViewModel;

set => ViewModel = (ViewModel)value;

}

public MainWindow()

{

InitializeComponent();

ViewModel = new ViewModel();

this.WhenActivated(disposable => //disposable para evitar memory leaks

{

//ViewModel: vm, v: View (MainWindow)

//Bind do Nome

this.Bind(ViewModel, vm => vm.Nome, v => v.LabelNomeShow.Content).DisposeWith(disposable);

//Bind do Email

this.Bind(ViewModel, vm => vm.Email, v => v.TBEmailSHow.Text).DisposeWith(disposable);

//Bind do Email (SIDE PANEL)

this.Bind(ViewModel, vm => vm.Email, v => v.SideEmail.Text).DisposeWith(disposable);

//Bind do Utilizador (SIDE PANEL)

this.Bind(ViewModel, vm => vm.Nome, v => v.SideHeader.Header).DisposeWith(disposable);

//Bind do Comando para abrir janela de adicionar lsita

this.BindCommand(ViewModel, vm => vm.CmdLogout, v => v.LogOut).DisposeWith(disposable);

//Bind do Comando para logOut

this.BindCommand(ViewModel, vm => vm.CmdOpenAddLista, v => v.BTNAdd).DisposeWith(disposable);

//Bind do Comando para editarPerfil

this.BindCommand(ViewModel, vm => vm.CmdOpenEditarPerfil, v => v.BTNEditarPerfil).DisposeWith(disposable);

//Bind do Comando para apagarPerfil

this.BindCommand(ViewModel, vm => vm.CmdEliminar, v => v.BTNEliminarPerfil).DisposeWith(disposable);

//Bind do Comando para mostar painel de perfil / conta

this.BindCommand(ViewModel, vm => vm.CmdShowConta, v => v.NavConta).DisposeWith(disposable);

//Bind do Comando para mostar painel de listas

this.BindCommand(ViewModel, vm => vm.CmdShowListasDash, v => v.NavListas).DisposeWith(disposable);

if (File.Exists("pfp.png"))

{

PFP.ImageSource = new BitmapImage(new Uri(Directory.GetCurrentDirectory() + "/pfp.png"));

}

else

{

PFP.ImageSource = new BitmapImage(new Uri("Images/user.png"));

}

});

ViewModel.CmdIsLogged

.Execute()

.Select(val => val ? true : false)

.Subscribe(val =>

{

if (val != true)

{

this.Hide();

Login dlg = new Login();

if (dlg.ShowDialog() == true)

{

ViewModel.CmdIsLogged.Execute().Subscribe();

LabelNomeShow.Content = ViewModel.Nome;

TBEmailSHow.Text = ViewModel.Email;

SideEmail.Text = ViewModel.Email;

SideHeader.Header = ViewModel.Nome;

this.Show();

}

else

{

this.Close();

}

}

else

{

LabelNomeShow.Content = ViewModel.Nome;

TBEmailSHow.Text = ViewModel.Email;

SideEmail.Text = ViewModel.Email;

SideHeader.Header = ViewModel.Nome;

this.Show();

}

});

ViewModel.CmdLogout

.Where(val => val == true)

.Subscribe(val =>

{

this.Hide(); Login dlg = new Login();

if (dlg.ShowDialog() == true) { this.Show(); }

else { this.Close(); }

});

ViewModel.CmdOpenAddLista

.Where(val => val == true)

.Subscribe(val =>

{

CriarLista dlg = new CriarLista("");

if (dlg.ShowDialog() == val)

{

ListasMain.Items.Clear();

listaSide.Items.Clear();

ViewModel.Listas.Clear();

ViewModel.CmdMostraListas.Execute().Subscribe();

foreach (var lista in ViewModel.Listas)

{

Button delBTN = new Button();

delBTN.Content = "Apagar";

delBTN.Width = 50;

delBTN.Margin = new Thickness(2);

delBTN.Name = lista.Key;

delBTN.HorizontalAlignment = HorizontalAlignment.Right;

delBTN.Click += DelBTN\_Click;

Button editBTN = new Button();

editBTN.Content = "Editar";

editBTN.Margin = new Thickness(2);

editBTN.Width = 50;

editBTN.Name = lista.Key;

editBTN.HorizontalAlignment = HorizontalAlignment.Right;

editBTN.Click += EditBTN\_Click;

CardViewItem CVI = new CardViewItem();

CVI.Header = lista.Key;

CVI.HorizontalAlignment = HorizontalAlignment.Right;

StackPanel SP = new StackPanel();

SP.Orientation = Orientation.Horizontal;

SP.HorizontalAlignment = HorizontalAlignment.Right;

SP.Children.Add(editBTN);

SP.Children.Add(delBTN);

CVI.Content = SP;

ListasMain.Items.Add(CVI);

listaSide.Items.Add(

new CardViewItem()

{ Header = lista.Key, Width = 140 });

}

}

});

ViewModel.CmdShowConta

.Where(val => val == true)

.Subscribe(val =>

{

conteudo.Visibility = Visibility.Hidden;

conta.Visibility = Visibility.Visible;

listaSide.Visibility = Visibility.Hidden;

contaSide.Visibility = Visibility.Visible;

LogOut.Visibility = Visibility.Visible;

circ1.Visibility = Visibility.Hidden;

circ2.Visibility = Visibility.Hidden;

});

ViewModel.CmdShowListasDash

.Where(val => val == true)

.Subscribe(val =>

{

conteudo.Visibility = Visibility.Visible;

conta.Visibility = Visibility.Hidden;

listaSide.Visibility = Visibility.Visible;

contaSide.Visibility = Visibility.Hidden;

LogOut.Visibility = Visibility.Hidden;

circ1.Visibility = Visibility.Visible;

circ2.Visibility = Visibility.Visible;

});

ViewModel.CmdMostraListas

.Execute()

.Where(val => val == true)

.Subscribe(val =>

{

foreach (var lista in ViewModel.Listas)

{

Button delBTN = new Button();

delBTN.Content = "Apagar";

delBTN.Width = 50;

delBTN.Margin = new Thickness(2);

delBTN.Name = lista.Key;

delBTN.HorizontalAlignment = HorizontalAlignment.Right;

delBTN.Click += DelBTN\_Click;

Button editBTN = new Button();

editBTN.Content = "Editar";

editBTN.Margin = new Thickness(2);

editBTN.Width = 50;

editBTN.Name = lista.Key;

editBTN.HorizontalAlignment = HorizontalAlignment.Right;

editBTN.Click += EditBTN\_Click;

CardViewItem CVI = new CardViewItem();

CVI.Header = lista.Key;

CVI.HorizontalAlignment = HorizontalAlignment.Right;

StackPanel SP = new StackPanel();

SP.Orientation = Orientation.Horizontal;

SP.HorizontalAlignment = HorizontalAlignment.Right;

SP.Children.Add(editBTN);

SP.Children.Add(delBTN);

CVI.Content = SP;

ListasMain.Items.Add(CVI);

listaSide.Items.Add(

new CardViewItem()

{ Header = lista.Key, Width = 140 });

}

});

ViewModel.CmdOpenEditarPerfil

.Where(val => val == true)

.Subscribe(val =>

{

ViewModel.CmdIsLogged.Execute().Subscribe();

EditarConta editarContadlg = new EditarConta(ViewModel.Nome, ViewModel.Email);

editarContadlg.ShowDialog();

ViewModel.CmdIsLogged.Execute().Subscribe();

LabelNomeShow.Content = ViewModel.Nome;

TBEmailSHow.Text = ViewModel.Email;

SideEmail.Text = ViewModel.Email;

SideHeader.Header = ViewModel.Nome;

});

ViewModel.CmdEliminar

.Where(val => val == true)

.Subscribe(val =>

{

this.Hide(); Login dlg = new Login();

if (dlg.ShowDialog() == true) { this.Show(); }

else { this.Close(); }

});

}

private void DelBTN\_Click(object sender, RoutedEventArgs e)

{

Button button = (Button)sender;

string name = button.Name;

ViewModel.LD = name;

if (MessageBox.Show("Deseja apagar a lista " + name + "?",

"Apagar Lista",

MessageBoxButton.YesNo,

MessageBoxImage.Question) == MessageBoxResult.Yes)

{

ViewModel.CmdDelLista

.Execute()

.Where(val => val == true)

.Subscribe(val =>

{

MessageBox.Show("Lista " + name + " apagada", "Lista Apagada", MessageBoxButton.OK, MessageBoxImage.Information);

listaSide.Items.Clear();

ListasMain.Items.Clear();

ViewModel.Listas.Clear();

ViewModel.CmdMostraListas.Execute().Subscribe();

foreach (var lista in ViewModel.Listas)

{

Button delBTN = new Button();

delBTN.Content = "Apagar";

delBTN.Width = 50;

delBTN.Margin = new Thickness(2);

delBTN.Name = lista.Key;

delBTN.HorizontalAlignment = HorizontalAlignment.Right;

delBTN.Click += DelBTN\_Click;

Button editBTN = new Button();

editBTN.Content = "Editar";

editBTN.Margin = new Thickness(2);

editBTN.Width = 50;

editBTN.Name = lista.Key;

editBTN.HorizontalAlignment = HorizontalAlignment.Right;

editBTN.Click += EditBTN\_Click;

CardViewItem CVI = new CardViewItem();

CVI.Header = lista.Key;

CVI.HorizontalAlignment = HorizontalAlignment.Right;

StackPanel SP = new StackPanel();

SP.Orientation = Orientation.Horizontal;

SP.HorizontalAlignment = HorizontalAlignment.Right;

SP.Children.Add(editBTN);

SP.Children.Add(delBTN);

CVI.Content = SP;

ListasMain.Items.Add(CVI);

listaSide.Items.Add(

new CardViewItem()

{ Header = lista.Key, Width = 140 });

}

});

}

}

private void EditBTN\_Click(object sender, RoutedEventArgs e)

{

Button button = (Button)sender;

string name = button.Name;

CriarLista dlg = new CriarLista(name);

if (dlg.ShowDialog() == true)

{

listaSide.Items.Clear();

ListasMain.Items.Clear();

ViewModel.Listas.Clear();

ViewModel.CmdMostraListas.Execute().Subscribe();

foreach (var lista in ViewModel.Listas)

{

Button delBTN = new Button();

delBTN.Content = "Apagar";

delBTN.Width = 50;

delBTN.Margin = new Thickness(2);

delBTN.Name = lista.Key;

delBTN.HorizontalAlignment = HorizontalAlignment.Right;

delBTN.Click += DelBTN\_Click;

Button editBTN = new Button();

editBTN.Content = "Editar";

editBTN.Margin = new Thickness(2);

editBTN.Width = 50;

editBTN.Name = lista.Key;

editBTN.HorizontalAlignment = HorizontalAlignment.Right;

editBTN.Click += EditBTN\_Click;

CardViewItem CVI = new CardViewItem();

CVI.Header = lista.Key;

CVI.HorizontalAlignment = HorizontalAlignment.Right;

StackPanel SP = new StackPanel();

SP.Orientation = Orientation.Horizontal;

SP.HorizontalAlignment = HorizontalAlignment.Right;

SP.Children.Add(editBTN);

SP.Children.Add(delBTN);

CVI.Content = SP;

ListasMain.Items.Add(CVI);

listaSide.Items.Add(

new CardViewItem()

{ Header = lista.Key, Width = 140 });

}

MessageBox.Show("Lista " + name + " alterada", "Lista Alterada", MessageBoxButton.OK, MessageBoxImage.Information);

}

}

}

}

## Xamarain

### RegisterPage.xaml.cs

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using Utad.Lab.G04.Domain.ViewModels;

using Xamarin.Forms;

using Xamarin.Forms.Internals;

using Xamarin.Forms.Xaml;

using ReactiveUI;

using System.Reactive.Disposables;

using System.Reactive.Linq;

namespace Utad.Lab.G04.MobileDroid.Views

{

/// <summary>

/// Page to sign in with user details.

/// </summary>

[Preserve(AllMembers = true)]

[XamlCompilation(XamlCompilationOptions.Compile)]

public partial class RegisterPage : IViewFor<ViewModel>

{

/// <summary>

/// Initializes a new instance of the <see cref="RegisterPage" /> class.

/// </summary>

///

public static readonly BindableProperty ViewModelProperty = BindableProperty

.Create(nameof(ViewModel), typeof(ViewModel), typeof(RegisterPage));

public ViewModel ViewModel

{

get => (ViewModel)GetValue(ViewModelProperty);

set => SetValue(ViewModelProperty, value);

}

object IViewFor.ViewModel

{

get => ViewModel;

set => ViewModel = (ViewModel)value;

}

public RegisterPage()

{

this.InitializeComponent();

ViewModel = new ViewModel();

this.WhenActivated(disposable => //disposable para evitar memory leaks

{

//vm: ViewModel, v: View (Login)

//Bind do Nome

this.Bind(ViewModel, vm => vm.Nome, v => v.NameEntry.Text).DisposeWith(disposable);

//Bind da Email

this.Bind(ViewModel, vm => vm.Email, v => v.EmailEntry.Text).DisposeWith(disposable);

this.Bind(ViewModel, vm => vm.Pais, v => v.PaisEntry.Text).DisposeWith(disposable);

//Bind da Password

this.Bind(ViewModel, vm => vm.Password, v => v.PasswordEntry.Text).DisposeWith(disposable);

this.Bind(ViewModel, vm => vm.PasswordConfirm, v => v.ConfirmPasswordEntry.Text).DisposeWith(disposable);

this.BindCommand(ViewModel, vm => vm.CmdRegister, v => v.btnRegister).DisposeWith(disposable);

this.OneWayBind(ViewModel, vm => vm.ErroCredenciais, v => v.ConfirmPasswordValidationLabel.Text).DisposeWith(disposable);

});

}

private async void BtRegister\_Clicked(object sender, System.EventArgs e)

{

ViewModel.CmdRegister

.Where(val => val != false)

.Subscribe(val => { Navigation.PushModalAsync(new NavigationPage(new LoginPage())); });

}

private async void BtX\_Clicked(object sender, System.EventArgs e)

{

await Navigation.PushModalAsync(new NavigationPage(new LoginPage()));

}

}

}

### LoginPage.xaml.cs

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using Utad.Lab.G04.Domain.ViewModels;

using Xamarin.Forms;

using Xamarin.Forms.Internals;

using Xamarin.Forms.Xaml;

using ReactiveUI;

using System.Reactive.Disposables;

using System.Reactive.Linq;

namespace Utad.Lab.G04.MobileDroid.Views

{

/// <summary>

/// Page to login with user name and password

/// </summary>

[Preserve(AllMembers = true)]

[XamlCompilation(XamlCompilationOptions.Compile)]

public partial class LoginPage : IViewFor<ViewModel>

{

/// <summary>

/// Initializes a new instance of the <see cref="LoginPage" /> class.

/// </summary>

public static readonly BindableProperty ViewModelProperty = BindableProperty

.Create(nameof(ViewModel), typeof(ViewModel), typeof(LoginPage));

public ViewModel ViewModel

{

get => (ViewModel)GetValue(ViewModelProperty);

set => SetValue(ViewModelProperty, value);

}

object IViewFor.ViewModel

{

get => ViewModel;

set => ViewModel = (ViewModel)value;

}

public LoginPage()

{

InitializeComponent();

ViewModel = new ViewModel();

this.WhenActivated(disposable => //disposable para evitar memory leaks

{

//vm: ViewModel, v: View (Login)

//Bind do Email

this.Bind(ViewModel, vm => vm.Email, v => v.EmailEntry.Text).DisposeWith(disposable);

//Bind da Password

this.Bind(ViewModel, vm => vm.Password, v => v.PasswordEntry.Text).DisposeWith(disposable);

//Bind do Erro: Apenas usa One Way

this.OneWayBind(ViewModel, vm => vm.ErroCredenciais, v => v.ValidationLabel.Text).DisposeWith(disposable);

//Bind do Comando para Login

this.BindCommand(ViewModel, vm => vm.CmdLogin, v => v.btnLogin).DisposeWith(disposable);

//Bind do Comando para Registo

this.BindCommand(ViewModel, vm => vm.CmdOpenRegisto, v => v.btnRegister).DisposeWith(disposable);

});

ViewModel.CmdOpenRegisto.Where(val => val != false).Subscribe(val =>

{ ViewModel.ErroCredenciais = ""; Navigation.PushModalAsync(new NavigationPage(new RegisterPage())); });

ViewModel.CmdLogin.Where(val => val == false).Subscribe(val => { Navigation.PushModalAsync(new NavigationPage(new PrimeiraPagina())); });

}

//private async void BtRegister\_Clicked(object sender, System.EventArgs e)

//{

// ViewModel.CmdOpenRegisto.Where(val => val != false).Subscribe(val =>

// { ViewModel.ErroCredenciais = ""; Navigation.PushModalAsync(new NavigationPage(new RegisterPage())); });

// //.Where(val => val != false)

// //.Subscribe(val =>

// //{

// //});

//}

//private async void BtLogin\_Clicked(object sender, EventArgs e)

//{

// ViewModel.CmdLogin.Where(val => val != false).Subscribe(val => { Navigation.PushModalAsync(new NavigationPage(new MainPage())); });

//}

}

}

### PrimeiraPagina.xaml.cs

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using Utad.Lab.G04.Domain.ViewModels;

using Xamarin.Forms;

using Xamarin.Forms.Internals;

using Xamarin.Forms.Xaml;

using ReactiveUI;

using System.Reactive.Disposables;

using System.Reactive.Linq;

namespace Utad.Lab.G04.MobileDroid.Views

{

/// <summary>

/// Page to show the setting.

/// </summary>

[Preserve(AllMembers = true)]

[XamlCompilation(XamlCompilationOptions.Compile)]

public partial class PrimeiraPagina : ContentPage, IViewFor<ViewModel>

{

/// <summary>

/// Initializes a new instance of the <see cref="PrimeiraPagina" /> class.

/// </summary>

public static readonly BindableProperty ViewModelProperty = BindableProperty

.Create(nameof(ViewModel), typeof(ViewModel), typeof(PrimeiraPagina));

public ViewModel ViewModel

{

get => (ViewModel)GetValue(ViewModelProperty);

set => SetValue(ViewModelProperty, value);

}

object IViewFor.ViewModel

{

get => ViewModel;

set => ViewModel = (ViewModel)value;

}

public PrimeiraPagina()

{

this.InitializeComponent();

ViewModel = new ViewModel();

this.WhenActivated(disposable => //disposable para evitar memory leaks

{

//vm: ViewModel, v: View (Login)

//Bind do Comando para Login

this.BindCommand(ViewModel, vm => vm.CmdOpenPerfil, v => v.btnPerfil).DisposeWith(disposable);

//Bind do Comando para Registo

this.BindCommand(ViewModel, vm => vm.CmdOpenListas, v => v.btnListas).DisposeWith(disposable);

this.BindCommand(ViewModel, vm => vm.CmdLogout, v => v.btnLogout).DisposeWith(disposable);

});

ViewModel.CmdLogout

.Where(val => val != true)

.Subscribe(val =>

{

Navigation.PushModalAsync(new NavigationPage(new LoginPage()));

});

ViewModel.CmdOpenEditarPerfil

.Where(val => val != true)

.Subscribe(val =>

{

Navigation.PushModalAsync(new NavigationPage(new ProfilePage()));

});

ViewModel.CmdOpenListas

.Where(val => val == true)

.Subscribe(val =>

{

Navigation.PushModalAsync(new NavigationPage(new ListasDeCompras()));

});

}

# Bibliografia

[1]

K. Bost, «5 Steps to Getting Started With Material Design In XAML», *IntelliTect*, 2 de julho de 2018. <https://intellitect.com/blog/getting-started-material-design-in-xaml/> (acedido 1 de maio de 2022).

[2]

ajcvickers, «Overview of Entity Framework Core - EF Core». <https://docs.microsoft.com/en-us/ef/core/> (acedido 31 de março de 2022).

[3]

«The Good and The Bad of Xamarin Mobile Development», *AltexSoft*. <https://www.altexsoft.com/blog/mobile/pros-and-cons-of-xamarin-vs-native/> (acedido 1 de maio de 2022).

[4]

davidbritch, «The Model-View-ViewModel Pattern - Xamarin». <https://docs.microsoft.com/en-us/xamarin/xamarin-forms/enterprise-application-patterns/mvvm> (acedido 29 de março de 2022).

[5]

«UX/UI & Brand design for Balance FarmTM by Igor Grytsiuk for Geex Arts on Dribbble». <https://dribbble.com/shots/17381040-UX-UI-Brand-design-for-Balance-Farm> (acedido 1 de maio de 2022).

[6]

«What is XAML? - The complete WPF tutorial». <https://wpf-tutorial.com/xaml/what-is-xaml/> (acedido 1 de maio de 2022).

[7]

«wpf Tutorial => The Model». <https://riptutorial.com/wpf/example/7013/the-model> (acedido 29 de março de 2022).

[8]

«Xamarin Forms  -  Usando Material Design». <https://www.macoratti.net/19/09/xf_matdesign1.htm> (acedido 1 de maio de 2022).

[9]

«Entendendo o Pattern Model View ViewModel MVVM», *DevMedia*. <https://www.devmedia.com.br/entendendo-o-pattern-model-view-viewmodel-mvvm/18411> (acedido 18 de junho de 2022).

[10]

«An advanced, composable, reactive model-view-viewmodel framework». <http://reactiveui.net/> (acedido 18 de junho de 2022).

[11]

BillWagner, «Language-Integrated Query (LINQ) (C#)». <https://docs.microsoft.com/en-us/dotnet/csharp/programming-guide/concepts/linq/> (acedido 18 de junho de 2022).

[12]

«Model–view–viewmodel», *Wikipedia*. 30 de maio de 2022. Acedido: 18 de junho de 2022. [Em linha]. Disponível em: <https://en.wikipedia.org/w/index.php?title=Model%E2%80%93view%E2%80%93viewmodel&oldid=1090651247>

[13]

L. Fujiwara, «ViewModels : A Simple Example», *Android Developers*, 16 de maio de 2019. <https://medium.com/androiddevelopers/viewmodels-a-simple-example-ed5ac416317e> (acedido 18 de junho de 2022).