UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ

NOME DO CURSO

Capa – elemento obrigatório

(apague todas as indicações em vermelho existentes neste modelo)

NOME COMPLETO DO AUTOR

**TÍTULO DO TRABALHO: SUBTÍTULO SE HOUVER**

TRABALHO DE CONCLUSÃO DE CURSO

CORNÉLIO PROCÓPIO

2016

NOME COMPLETO DO AUTOR

Folha de rosto – elemento obrigatório

**TÍTULO DO TRABALHO: SUBTÍTULO SE HOUVER**

Trabalho de Conclusão de Curso de graduação, apresentado à disciplina Nome da Disciplina, do curso de Nome do Curso da Universidade Tecnológica Federal do Paraná – UTFPR, como requisito parcial para a obtenção do título de Bacharel.

Orientador: Prof. Dr. Nome Sobrenome

CORNÉLIO PROCÓPIO

2016

**FOLHA DE APROVAÇÃO**

Utilizar o modelo utilizado pelo curso – elemento obrigatório

“A Folha de Aprovação assinada encontra-se na Coordenação do Curso”

Dedicatória – elemento opcional

Dedico este trabalho à minha família.

**AGRADECIMENTOS**

Elemento opcional

Agradeço ao meu orientador Prof. Dr. Nome do Orientador, pela sabedoria com que me guiou nesta trajetória.

Aos meus colegas de sala.

A Secretaria do Curso, pela cooperação.

Gostaria de deixar registrado também, o meu reconhecimento à minha família, pois acredito que sem o apoio deles seria muito difícil vencer esse desafio.

Enfim, a todos os que por algum motivo contribuíram para a realização desta pesquisa.

Epígrafe – elemento opcional

Espaço destinado à epígrafe (elemento opcional). Nesta folha, o autor usa uma citação, seguida de indicação de autoria e ano, relacionada com a matéria tratada no corpo do trabalho.

**RESUMO**

Elemento obrigatório

SOBRENOME, Prenome do Autor do Trabalho. **Título do trabalho:** subtítulo (se houver). 2014. 65 f. Trabalho de Conclusão de Curso (Graduação) – Engenharia de Software. Universidade Tecnológica Federal do Paraná. Cornélio Procópio, 2014.

Elemento obrigatório, constituído de uma sequência de frases concisas e objetivas, fornecendo uma visão rápida e clara do conteúdo do estudo. O texto deverá conter no máximo 500 palavras e ser antecedido pela referência do estudo. Também, não deve conter citações. O resumo deve ser redigido em parágrafo único, espaçamento simples e seguido das palavras representativas do conteúdo do estudo, isto é, palavras-chave, em número de três a cinco, separadas entre si por ponto e finalizadas também por ponto. Usar o verbo na terceira pessoa do singular, com linguagem impessoal (pronome SE), bem como fazer uso, preferencialmente, da voz ativa.

**Palavras-chave:** Primeira palavra. Segunda palavra. Terceira palavra. Quarta pala-vra. Quinta-palavra.

**ABSTRACT**

Elemento obrigatório

SOBRENOME, Prenome do Autor do Trabalho. **Title of the working:** subtitle (if any). 2014. 65 f. Trabalho de Conclusão de Curso (Graduação) – Engenharia de Software. Universidade Tecnológica Federal do Paraná. Cornélio Procópio, 2014.

Elemento obrigatório, constituído de uma sequência de frases concisas e objetivas, fornecendo uma visão rápida e clara do conteúdo do estudo. O texto deverá conter no máximo 500 palavras e ser antecedido pela referência do estudo. Também, não deve conter citações. O abstract deve ser redigido em idioma de divulgação internacional (sugere-se em inglês, em outras línguas, que não o inglês, consultar o departamento/curso de origem), em parágrafo único, espaçamento simples e seguido das palavras representativas do conteúdo do estudo, isto é, palavras-chave, em número de três a cinco, separadas entre si por ponto e finalizadas também por ponto. Usar o verbo na terceira pessoa do singular, com linguagem impessoal (pronome SE), bem como fazer uso, preferencialmente, da voz ativa.

**Keywords**: Keyword. Keyword. Keyword. Keyword. Keyword.

**LISTA DE GRÁFICOS**

Elemento opcional

|  |  |
| --- | --- |
| GRÁFICO 1 – CULTURA DA INFORMAÇÃO NAS INSTITUIÇÕES................... | 80 |
| GRÁFICO 2 – OBTENÇÃO E COLETA DE INFORMAÇÕES............................. | 82 |
| GRÁFICO 3 – UTILIZAÇÃO DE SISTEMAS EM TEMPO REAL E A DISPONI-BILIDADE DE COMPUTADORES................................................ | 86 |
| GRÁFICO 4 – FLUXO E COMPARTILHAMENTO DE INFORMAÇÕES E CO-NHECIMENTO.............................................................................. | 92 |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |

**LISTA DE TABELAS**

Elemento opcional

|  |  |
| --- | --- |
| TABELA 1 - INSTITUIÇÕES DE EDUCAÇÃO SUPERIOR (IES) POR ORGA-NIZAÇÃO ACADÊMICA - 2004......................................................... | 20 |
| TABELA 2 - SITUAÇÃO DA EDUCAÇÃO BRASILEIRA EM 2002...................... | 20 |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |

**LISTA DE ABREVIATURAS**

Elemento opcional

|  |  |
| --- | --- |
| acep. | Acepção |
| Biol. | Biologia |
| flex. | Flexão |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |

**LISTA DE SIGLAS**

Elemento opcional

|  |  |
| --- | --- |
| ABIPTI | Associação Brasileira das Instituições de Pesquisa Tecnológica |
| ANPEI | Associação Nacional de Pesquisa, Desenvolvimento e Engenharia das Empresas Inovadoras |
| APO | Administração por Objetivos |
| BSC | Balanced Scorecard |
| CE | Capital Estrutural |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |

**LISTA DE ACRÔNIMOS**

Elemento opcional

|  |  |
| --- | --- |
| CAE | Computer Aided Engineering |
| FORTRAN | Formula Translation |
| IPPUC | Instituto de Pesquisa e Planejamento Urbano de Curitiba |
| NASA | National Aeronautics and Space Administration |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |

**LISTA DE SÍMBOLOS**

Elemento opcional

|  |  |
| --- | --- |
| Υ | Coeficiente de atenuação da função da onda do elétron no óxido |
| λ | Parâmetro de ajuste para modulação do comprimento do canal |
| σ | Efeito DIBL |
| Ө | Parâmetro de entrada para ajuste de mobilidade |
| έ | Permissividade elétrica do silício |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |

**SUMÁRIO**

Elemento obrigatório

|  |  |  |
| --- | --- | --- |
| **1** | **INTRODUÇÃO.......................................................................................** | **15** |
| **1.1** | **Delimitação do tema............................................................................** | **15** |
| **1.2** | **Problemas e premissas.......................................................................** | **15** |
| **1.3** | **Objetivos...............................................................................................** | **16** |
| 1.3.1 | Objetivo geral......................................................................................... | 16 |
| 1.3.2 | Objetivos específicos............................................................................. | 16 |
| **1.4** | **Justificativa..........................................................................................** | **16** |
| **1.5** | **Procedimentos metodológicos..........................................................** | **16** |
| **1.6** | **Referencial teórico...............................................................................** | **16** |
| **1.7** | **Estrutura...............................................................................................** | **16** |
| **2** | **ASSUNTO INICIAL...............................................................................** | **17** |
| **2.1** | **Desdobramento do assunto inicial....................................................** | **17** |
| 2.1.1 | Abordagem específica do assunto inicial.............................................. | 18 |
| 2.1.1.1 | Tabelas.................................................................................................. | 19 |
| 2.1.1.2 | Equações............................................................................................... | 21 |
| **3** | **TEMA SEGUINTE.................................................................................** | **21** |
| **3.1** | **Primeira abordagem do tema seguinte..............................................** | **21** |
| **3.2** | **Segunda abordagem do tema seguinte.............................................** | **21** |
| 3.2.1 | Primeira especificação........................................................................... | 21 |
| **4** | **RESULTADOS OBTIDOS.....................................................................** | **21** |
| **5** | **CONSIDERAÇÕES FINAIS..................................................................** | **22** |
|  | **REFERÊNCIAS.....................................................................................** | **23** |
|  | **APÊNDICE A – LEITURA DO ARQUIVO XML PARA CONFIGURAÇÃO DO SISTEMA..........................................................** | **24** |
|  | **APÊNDICE B – MODELOS DE CASOS DE TESTE............................** | **26** |
|  | **ANEXO A – REGULAMENTO TÉCNICO PARA INSPEÇÃO SANITÁRIA DE ALIMENTOS...............................................................** | **28** |

1 INTRODUÇÃO

Parte inicial do texto, na qual devem constar o tema e a delimitação do assunto tratado, objetivos da pesquisa e outros elementos necessários para situar o tema do trabalho, tais como: justificativa, procedimentos metodológicos (classificação inicial), embasamento teórico (principais bases sintetizadas) e estrutura do trabalho, tratados de forma sucinta.

Salienta-se que os procedimentos metodológicos e o embasamento teórico são tratados, posteriormente, em capítulos próprios e com a profundidade necessária ao trabalho de pesquisa.

1.1 Delimitação do tema

1.2 Problemas e premissas

1.3 Objetivos

2 **FUNDAMENTAÇÃO TEÓRICA**

...

Após a introdução do trabalho, da apresentação do tema, dos problemas e dos objetivos a sequencia é a descrição dos assuntos propostos, com uma exposição ordenada e pormenorizada.

A composição dos assuntos pode ser realizada em seções e subseções, material e método(s) e/ou metodologia e resultados, agora descritos detalhadamente. Cada seção ou subseção deverá ter um título apropriado ao conteúdo. A orientação é que seja utilizada a terceira pessoa do singular na elaboração do texto, mantendo-se a forma impessoal no mesmo. (UNIVERSIDADE..., 2009, p. 38).

2.x Aplicações Móveis

As plataformas móveis possuem três classificações, aplicações nativas, aplicações Web e aplicações híbridas[ref]:

1. Aplicações navitvas: Aplicações nativas são programas que são desenvolvidos para uma plataforma específica. A forma de desenvolvimento é definida pala organização proprietária. Esta empresa fornece para os desenvolvedores o *Software Development Kit*(SDK)e uma *Integrated Development Environment*(IDE), que é o conjunto de ferramentas necessárias para desenvolver um aplicativo para a aquela plataforma. Ao optar por uma aplicação nativa o desenvolvedor tem acesso a todas as *Application Programming Interface* (API) para aquela plataforma sem restrições, podendo ser utilizadas no momento em que estiverem disponíveis para a comunidade de programadores.

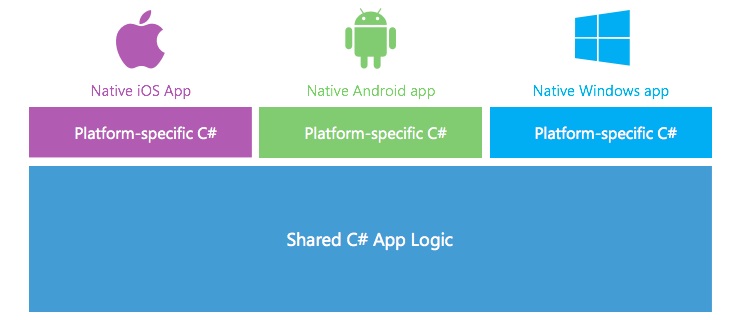
Tabela I

2.1 Xamarin

Xamarin é um *framework* de desenvolvimento para dispositivos móveis *cross-plataform* e mantido pela Microsoft. O *framework* Xamarin possui três técnicas diferentes para criar aplicações móveis: Xamarin.iOS, Xamarin.Android e Xamarin.Forms.

2.2 Xamarin.iOs e Xamarin.Android

Ambos são criados sobre o Mono, uma versão de software livre do .NET Framework. Os aplicativos feitos em uma dessas duas técnicas, ficam limitados a plataforma escolhida.Por exemplo, caso o desenvolvedor escolha o Xamarin.Android, nenhuma tela que foi criada para interação do usuário será reaproveitada quando o time de desenvolvimento decidir fazer o mesmo aplicativo para iOS, obrigando o os desenvolvedores a recriar toda a interface e a lógica de apresentação, reaproveitado apenas a lógica de aplicação, como pode ser visto na Figura 1.



**Figura 1: Arquitetura de desenvolvimento de aplicativos Xamarin.Android e Xamarin.iOS**

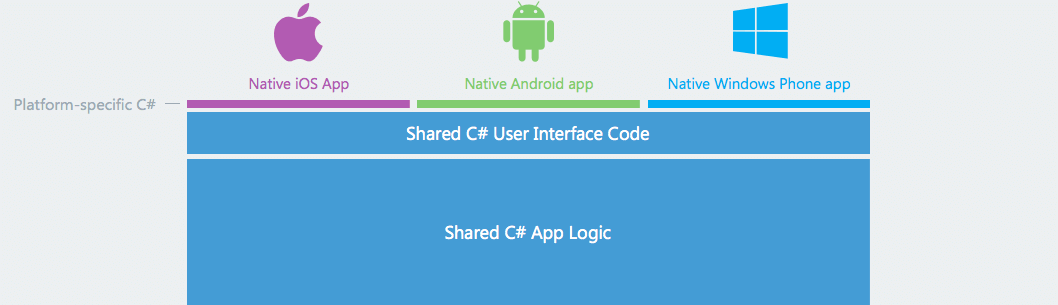
**Fonte: (site)**

Na figura 1, é possível notar que todo o código da lógica da aplicação é escrito em C# e utilizado em todas as plataformas (Shared C# App Logic), contudo a para cada uma das plataformas será necessário recriar a lógica de apresentação (Plataform-specific C#).

Os aplicativos são escritos na linguagem C# e compilados e utilizando a versão específica da DLL(Dynamic-link Library) de acordo com a plataforma. No caso, MonoTouch.dll para iOS e MonoAndroid.dll para Android. Após o processo de compilação o resultado é um pacote de aplicação idêntico aos que são feitos nas IDEs padrão, sendo impossível distinguir um .apk feito em Android Studio e outro feito em Xamarin.

2.3 Xamarin.Forms

O Xamarin.Forms é uma abstração da forma de criar aplicações móveis, utilizando essa tecnologia é possível escrever um único código que será interpretado e compilado individualmente em cada plataforma. As interfaces de usuário são renderizadas e transformadas em controles nativos. Diferente do Xamarin.Android e Xamarin.iOS que apenas a lógica de aplicação é compartilhada, no Xamarin.Forms tanto a interface quanto a própria lógica de aplicação são escritas uma única vez. Quando um aplicativo é compilado ele utiliza a API nativa de cada plataforma. O Xamarin.Forms visa trazer agilidade para os times de desenvolvimento, sendo necessário que todos conheçam apenas uma linguagem de programação. Além disso, caso tenha algum problema ou o time decida fazer alguma melhoria, basta atualizar apenas uma base de código. Como é mostrado na Figura 2, tanto a lógica de aplicação (Shared C# App Logic) quanto a lógica de apresentação e interface (Shared C# User Interface Code).



**Figura 2: Arquitetura de desenvolvimento de aplicativos Xamarin.Forms**

**Fonte (site)**

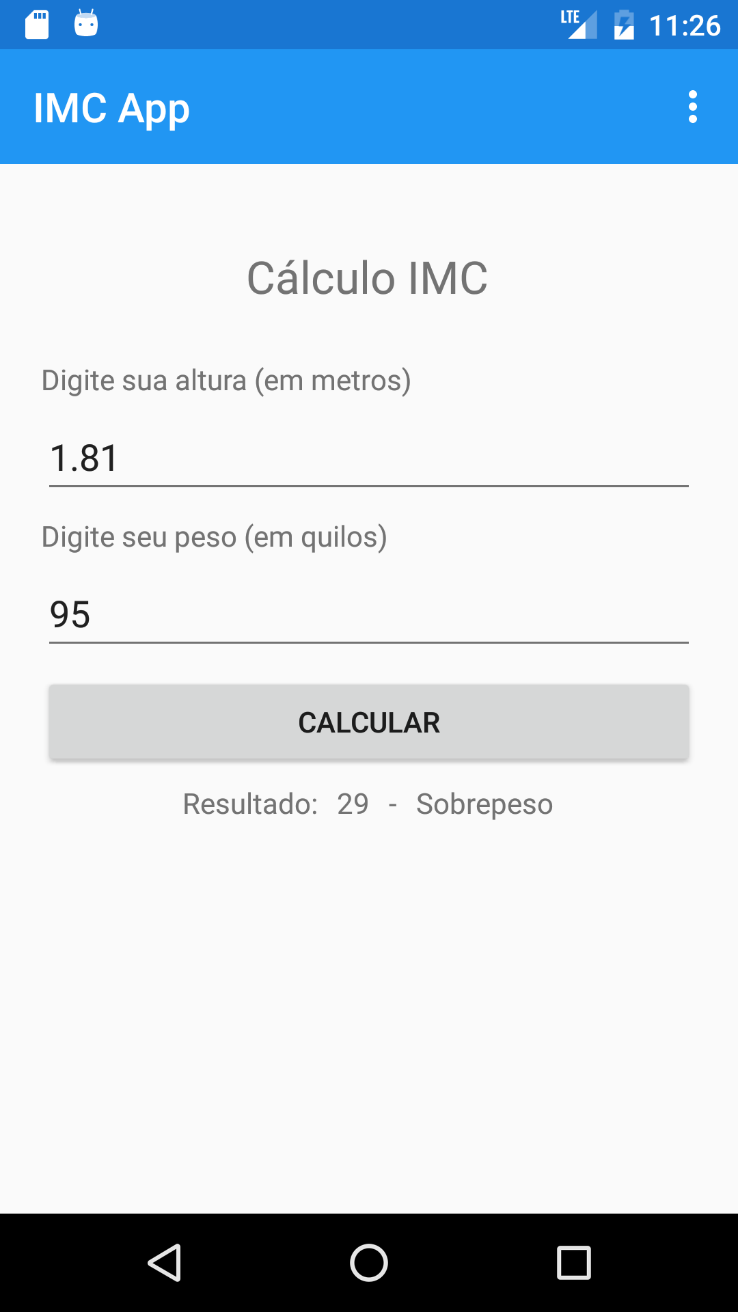
Para exemplificar a utilização do Xamarin.Forms será utilizado um projeto de cálculo de Índice de Massa Corporal (IMC). O IMC é o resultado de uma fórmula matemática que indica como a sua saúde em relação a sua massa corporal, contudo esse índice é apenas um ponto de partida, já que o IMC não define exatamente o seu estado nutricional. A fórmula para realizar o cálculo é: IMC = P(peso em quilos)/A²(altura x altura, em metros). Lembrando que o IMC somente é válido para pessoas adultas e com idades entre 20 a 59 anos. A aplicação desenvolvida é composta de apenas uma tela que possuem duas entradas de dados, a primeira é a altura que deve ser inserida pela medida de uma pessoa adulta, e em metros. A próxima entrada é o peso que deve ser informado em quilogramas. Os dois controles aceitam somente números que podem conter casas decimais. Logo abaixo existe o botão “CALCULAR”, que chamará o método da fórmula do cálculo e passará os dois valores como parâmetro. Caso nenhum dos campos tenham valores em branco, nulos ou zeros, ao clicar no botão o aplicativo exibirá o resultado do cálculo e um texto informando em qual das faixas de peso a pessoa está enquadrada. Os possíveis resultados são:

Baixo peso: Caso o resultado seja menor que 18,5.

Peso adequado: Caso o resultado seja maior ou igual a 18,5 e menor que 25.

Sobrepeso: Caso o resultado seja maior ou igual a 25 e menor que 30.

Obesidade: Caso o resultado seja maior ou igual a 30.



**Figura 3: Aplicativo exemplo – Tela inicial**

**Fonte: Autoria própria**

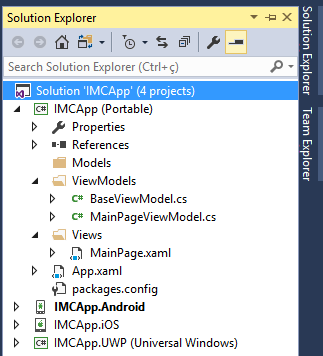
1.4 Estrutura do Projeto

A IDE necessária para codificar uma aplicação Xamarin é o Xamarin Studio no Mac e o Visual Studio no Windows. Para desenvolver este exemplo foi utilizado um computador com Windows 10 e o Visual Studio na versão 2017.

Um projeto Xamarin.Forms é um conjunto de projetos, denominado Solution e é composto por no mínimo quatro projetos: Um Portable, um Android, um iOS e outro UWP. Todo o código será escrito no projeto Portable e quando for executar a saída será necessário compilar cada projeto individualmente para a plataforma desejada. Caso seja necessário utilizar algum recurso muito específico de alguma das plataformas será necessário criar um código dentro do projeto de cada uma delas, já que o que não estiver na Portable não será compartilhado com os outros projetos.

O projeto foi desenvolvido utilizando o padrão Model View ViewModel (MVVM), criado pela Microsoft e era anteriormente utilizado pelo WPF e Silverlight. Este padrão visa separar as responsabilidades dos objetos. A View é a interface do usuário e a sua única função é exibir os controles, como botões ou textos. A Model é onde fica toda a lógica de negócio e os dados, os métodos de acesso a banco de dados geralmente estão nas Models. A ViewModel faz a ligação entre esses dois objetos, afinal a View e a Model não estão relacionadas diretamente, não sendo possível. Na ViewModel é programada toda a lógica de apresentação. A comunicação com a View é feita através dos databindings. Ao escrever o XAML da View, pode-se associar um controle da interface à uma propriedade da ViewModel, com o databinding, a vantagem disso é que qualquer alteração que for feita em qualquer um dos arquivos, seja por interação do usuário na interface ou por algum cálculo feito internamente na ViewModel, a propriedade será notificada e alterada, não sendo necessário utilizar eventos para monitorar as ações do usuário. Caso o time de desenvolvimento seja bem separado e bem definido, ainda é possível que a View e a ViewModel sejam programadas por pessoas ou times diferentes, um deles cuidando apenas do XAML da View e outro criando os métodos na classe C# da ViewModel, basta que apenas o time que criará a View descreva em detalhes o que será exibido ao usuário e qual o seu comportamento. Uma outra vantagem de ter toda a lógica de apresentação na ViewModel é que isso torna o código completamente testável, uma vez que uma ViewModel nada mais é do que uma simples classe e totalmente compatível com qualquer *framework* de testes automatizados que suportem a linguagem C#.

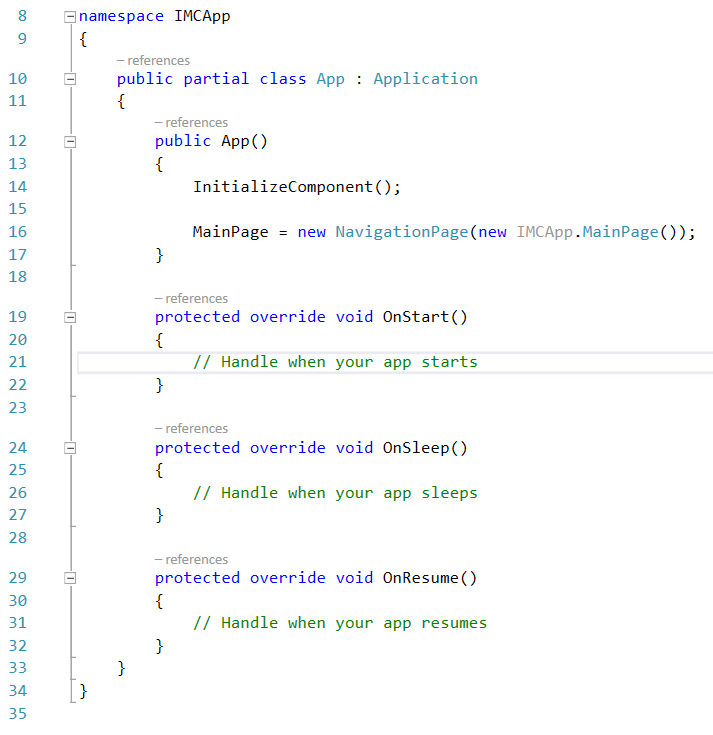
É uma boa prática de programação dividir as responsabilidades em pastas, a título de organização. Como é possível ver na imagem a seguir as ViewModels do projeto são classes C# e cada View é um arquivo com extensão .XAML e que possui também uma classe C# relacionada a View, chamada de *code-behind*. No *code-behind* também é possível escrever toda a lógica de apresentação, contudo isso cria um acoplamento muito forte entre a View e o seu respectivo *code-behind*, caso em algum momento o time de desenvolvimento precise remover uma View ou alterar a sua ordem ou lógica de apresentação, será necessário refatorar todo o código, procurando por dependências. Já a ViewModel que como possui um baixo acoplamento com as Views podem ser separadas a qualquer momento. Nesse exemplo temos a View MainPage.xaml e MainPage.xaml.cs que é o seu *code-behind*. Por questões de organização as ViewModels que contém a lógica de apresentação da View, recebem o mesmo nome e no final acrescenta-se “ViewModel”, como no exemplo temos a MainPage o nome da nossa ViewModel será MainPageViewModel.cs Existe também a BaseViewModel; essa classe tem a abstração de alguns métodos ou atributos que serão utilizados em todas as outras ViewModels.

****

**Figura 4: Estrutura de arquivos do projeto**

**Fonte: Autoria própria**

O arquivo App.xaml apesar de ter as mesmas propriedades de uma view, como a extensão XAML e o *code-behind*, ela é apenas um objeto que extrai os métodos de ciclo de vida dos aplicativos, como é possível ver na figura 5:

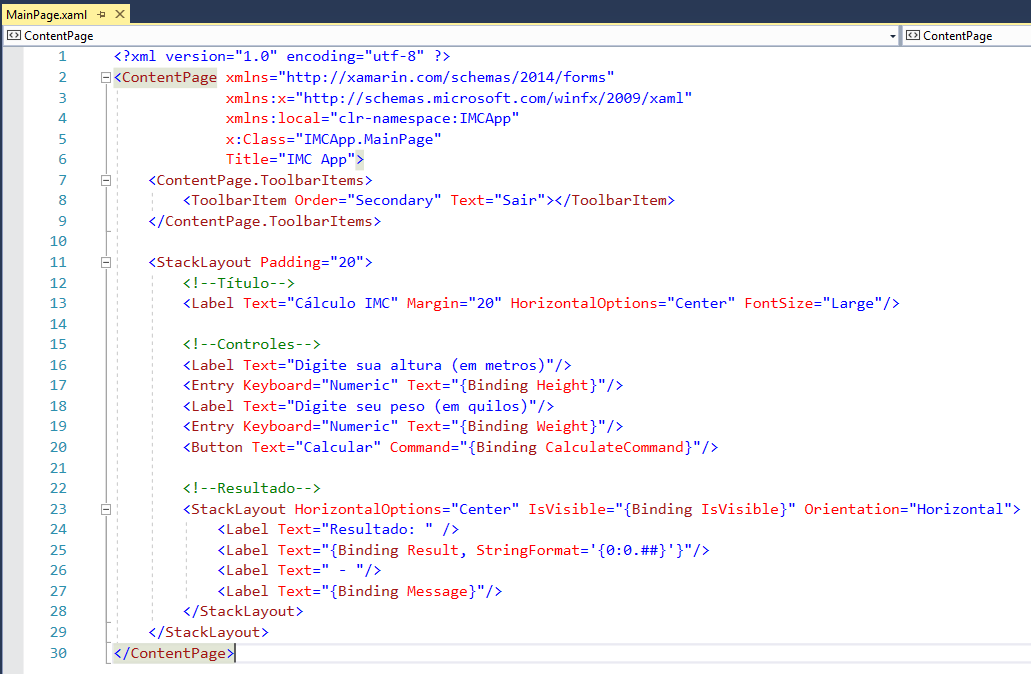


**Figura 5: Trecho de código da classe App.xaml.cs**

**Fonte: Autoria própria**

Através dos métodos da classe App.xaml.cs é possível controlar os comportamentos de ciclo de vida da aplicação em cada uma das plataformas. Caso seja inserida alguma lógica de apresentação no método *OnStart()*, está será a primeira coisa a ser executada assim que o aplicativo for iniciado. No *OnSleep()* a programação só terá efeito quando o aplicativo for colocado em segundo plano, quando o usuário navegar por outro aplicativo por exemplo. Já o *OnResume()* só será executado quando o aplicativo voltar a ficar ativo, em primeiro plano.

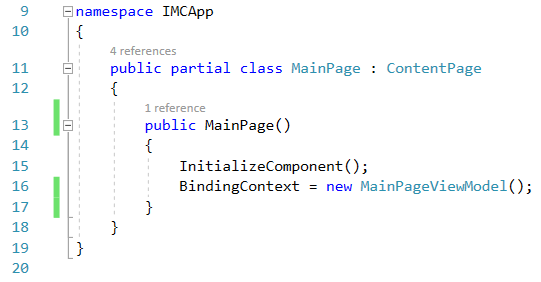
É na App.xaml.cs que a aplicação é iniciada, onde é instanciado o primeiro objeto, no método construtor da classe. MainPage é a primeira página da aplicação e no caso atribuímos uma NavigationPage e passando MainPage como parâmetro. Utilizando o NavigationPage é permitido fazer uso da navegação por pilha, presente em todas as plataformas *mobile*, onde a cada nova tela aberta o SO empilha a nova tela sobre a anterior e quando o usuário realiza a ação para voltar, ele desempilha e destrói aquela tela, podendo desempilhar até chegar a tela inicial, a MainPage.

****

**Figura 6: Trecho de código da View MainPage.xaml**

**Fonte: Autoria própria**

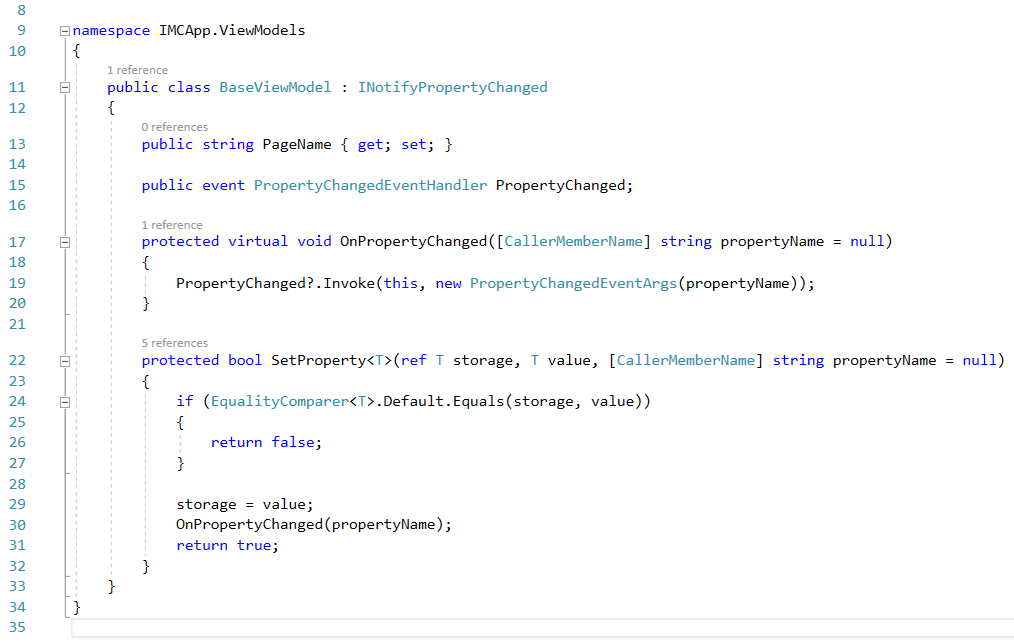
Na figura 6 é mostrado o conteúdo de um arquivo XAML. Toda a interface do usuário é escrita neste arquivo, utilizando da abstração do Xamarin.Forms ao inserir a tag *<Button>*, na linha 20, o Xamarin transformará em tempo de compilação essa *tag* em um controle, invocando a *API* nativa referente a um botão, de acordo com a plataforma para qual estiver desenvolvendo. É possível notar que alguns controles, como os as entradas de texto das linhas 17 e 19, foi utilizado o conceito de *databindings*, presente no Xamarin.Forms. Com isso ao inserir ou alterar um valor novo no campo, como ele está ligado a uma propriedade de uma classe, não é necessário monitorar as alterações desse controle, já que todas as mudanças serão automaticamente refletidas na classe. Quase todos os atributos de uma *tag* podem ser definidas por *databindings*, não se limitando apenas a textos ou números. Por exemplo o controle de interface *StackLayout*, da linha 23, ele é responsável por agrupar elementos de interface. Ele possui uma propriedade *IsVisible* da qual o valor está associado a uma propriedade booleana da classe, podendo receber verdadeiro ou falso, ou seja, dependendo da lógica de apresentação, o *StackLayout* pode ou não estar visível. Neste exemplo do IMC se os valores de altura (*Height*) ou peso (*Weight*) forem igual a zero e o usuário clicar no botão para fazer o cálculo, a propriedade *IsVisible* recebe o valor *false* e o elemento deixa de ser exibido na tela.



**Figura 6: Trecho de código da classe MainPage.xaml.cs**

**Fonte: Autoria própria**

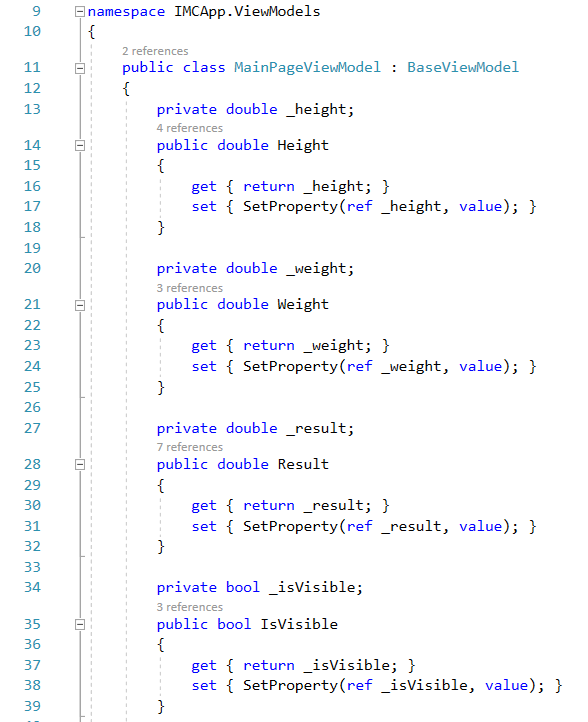
A figura 7 contém todo o código do *code-behind* da View MainPage.xaml. Como toda a lógica de apresentação deve estar contida em uma ViewModel, a única instrução que a classe executa é informar qual é o seu *BindingContext* na linha 16, ou seja, qual a classe que a View deve associar as suas propriedades.

****

**Figura 8: trecho de código da classe BaseViewModel.cs**

**Fonte: Autoria própria**

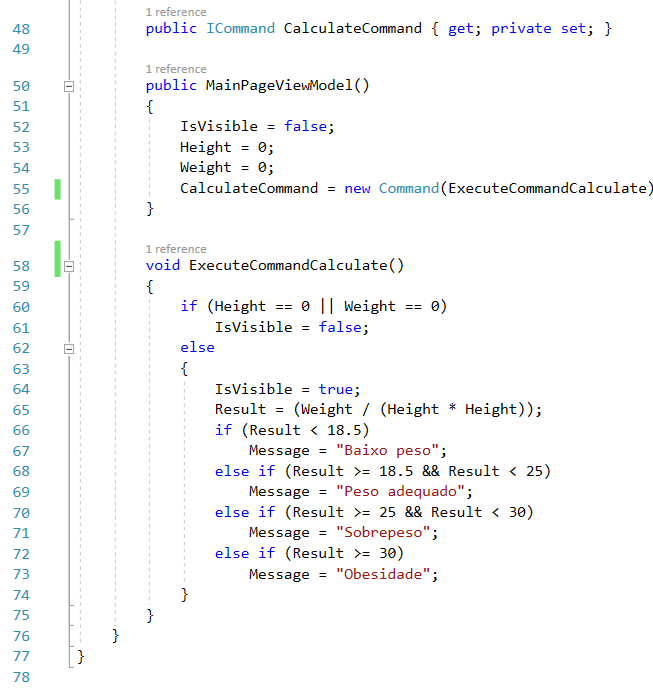
Na figura 8 é mostrado o código da BaseViewModel.cs, uma classe onde abstrai os métodos e propriedades que serão utilizados nas demais ViewModels. Para tirar trabalhar com os *databindings* é necessário que seja implementada a interface *INotifyPropertyChanged,* na linha 11. Nessa interface temos o evento *PropertyChangedEcentHandler*, na linha 15 que junto com o método *OnPropertyChanged(),* na linha 17, é responsável por monitorar as propriedades e informar quando elas foram alteradas. Já na linha 22 temos o método *SetProperty()*, que deve ser incluído nos métodos *setters* das propriedades das ViewModels. Nele é executada a verificação para garantir que haja a mudança do valor apenas quando a propriedade for alterada, economizando processamento desnecessário do dispositivo.

****

**Figura 9: Trecho de código da classe MainPageViewModel.cs**

**Fonte: Autoria própria**

No trecho de código da figura 9 a classe MainPageViewModel.cs implementa a BaseViewModel.cs. Quando definimos as propriedades da classe, todas as que terão alguma alteração na exibição para o usuário devem chamar o método *SetProperty()* nos seus métodos *setters.* Só assim o Xamarin garante que o valor será sempre o mesmo, em qualquer um dos arquivos, sem que seja necessário criar um evento na interface que verifique se os valores foram alterados. Todas essas propriedades estão ligadas a um controle na View.



**Figura 10: Trecho de código da classe MainPageViewModel.cs**

**Fonte: Autoria própria**

A figura 10 exibe toda a lógica de apresentação da MainPageViewModel.cs para a MainPage.xaml. No construtor são definidos os parâmetros inicias para a execução da aplicação. A propriedade *IsVisible* é iniciada com falso, ocultando o controle de *layout* da View, já que no início da aplicação nenhum cálculo foi executado e sendo assim não há nada a ser exibido. O *CalculateCommand* é a propriedade do tipo *Command*, presente *framework* Xamarin.Forms, que é associado a um método o *ExecuteCommandCalculate()* e é nele onde é executado a lógica para a ação do botão “CALCULAR” da interface, que também está ligado a classe por *databindings*.

O método verifica primeiramente se as propriedades *Height* e *Weight* estão com valor zero, se verdadeiro, *IsVisible* recebe *false* e o método termina, caso contrário o valor a ser atribuído será *true* e os valores de altura e peso serão calculados na fórmula. Para cada um dos valores existe uma saída em texto correspondente que será atribuído à propriedade *Message*, que está contida no controle *StackLayout.*

5 CONSIDERAÇÕES FINAIS

Parte final do texto, na qual se apresentam as conclusões do trabalho acadêmico, usualmente denominada considerações finais. Pode ser usada outra denominação similar que indique a conclusão do trabalho.

**REFERÊNCIAS**

OLIVEIRA NETTO, A. A. de. **Metodologia da pesquisa científica:** guia prático para a apresentação de trabalhos acadêmicos. 3. ed. rev. e atual. Florianópolis: Visual Books, 2008.

UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ. Sistema de Bibliotecas. **Normas para elaboração de trabalhos acadêmicos.** Curitiba: UTFPR, 2009.

COSTA, V. R. À margem da lei. **Em Pauta,** Rio de Janeiro, v. 2, n. 18, p. 12-25, set. 1998. Disponível em: <www.google.com.br>. Acesso em: 13 jan. 2015.

FIGUEIREDO, K. M. de. **Mapeamento dos modos de transferência metálica.** 2000. 75 f. Dissertação (Engenharia Elétrica) – Universidade Federal do Paraná, Curitiba, 2000.

FURTADO, A. T.; CARVALHO, R. de. Padrões de intensidade. In: SIMPÓSIO INTERNACIONAL DE GESTÃO, 4., 2001, Curitiba. **Anais...** São Paulo: USP, 1994. p. 15-28.