06

Modelo MVVM

Transcrição

[00:00] O que eu quero fazer agora? Eu quero trazer os dados reais que estão armazenados no banco, só que antes disso eu quero refatorar a nossa aplicação para eu colocar essa página, essa AgendamentosUsuarioView, no padrão “mvvm”, ainda não tenho a ViewModel para essa página.

[00:23] Eu vou copiar esse nome e vou lá no ViewModel, na pasta ViewModel e vou adicionar uma nova classe, que eu vou dar o nome de “AgendamentosUsuarioViewModel”, para deixar no padrão, com isso eu já tenho uma ViewModel para poder trabalhar, o que eu vou fazer agora é mover essa lista, mesmo essa lista falsa de dados aqui, eu vou mover lá para o nosso ViewModel.

[00:54] Primeiro eu vou declarar na ViewModel uma propriedade, eu vou chamar de “lista”, essa propriedade vai ser uma lista de agendamentos, “list”, vou importar a referência e vou chamar essa lista de “Lista” mesmo, essa lista vai levar para View quais são os agendamentos realizados.

[01:25] Eu vou definir aqui dentro do “get”, e vou colocar aqui o código vai retornar a lista de agendamentos de realizados, vou copiar esse código aqui “ItemsSource” e eu vou trazer esse “new List” lá para dentro do ViewModel, eu vou retornar a mesma lista, eu coloco o “set” como privado, “private set”, vou formatar o documento, eu vou fazer aqui “this.BindingContext” e vou associar com uma nova instância da nossa ViewModel.

[02:11] Eu vou pegar o nome da ViewModel, aqui o “BindingContext” é igual a uma nova, AgendamentosUsuarioViewModel, vou importar a referência, agora eu removo esse código que ficou sobrando aqui embaixo.

[02:30] Por último, falta mexermos lá no “xaml”, a propriedade “ItemsSource”, que vai vir através de um “binding”, eu coloco aqui o “Binding Lista”, nessa ListView tínhamos definido o nome, só para podermos ter um ponto de acesso, para poder identificar ela a partir do Code Behind, agora como eu não tenho mais esse acesso via Code Behind, eu posso simplesmente remover o nome, abrimos a ViewModel, “AgendamentosUsuarioViewModel”.

[03:07] Aqui temos essa lista, que é o do tipo List, de agendamento, vamos trocar, em vez do List, vamos utilizar um “ObservableCollection”, com isso eu teria que modificar o tipo de retorno como uma nova “ObservableCollection” também, mas como não vamos utilizar esse código, vamos trazer do banco de dados, eu vou simplesmente remover essas linhas aqui, ficamos com uma propriedade de vazia, essa propriedade “Lista”.

[03:50] Eu vou criar aqui um atributo novo, vai ser um atributo local, eu vou chamar de “lista” com minúscula e essa lista vai ser manipulada dentro dessa propriedade, no retorno eu quero retornar a lista que é a lista local e no “set” eu quero definir, eu quero atribuir também essa lista local, “lista = value”, eu também vou utilizar o que?

[04:16] Eu vou usar o nosso método “OnPropertyChanged” que vai fazer com que essa propriedade seja notificada sempre que ela foi atualizada, eu faço aqui “OnPropertyChanged”, você percebe que esse nome não aparece na IntelliSense, esse método não existe para essa classe, é claro, porque essa classe é uma classe simples, ela é vazia, ela só tem essa propriedade, o que eu preciso fazer é herdar da nossa Base Model, que já foi utilizado em vários momentos, “BaseViewModel”, vou importar a referência.

[04:56] Agora eu consigo notificar, agora o que eu quero fazer é inicializar essa lista assim que a ViewModel for instanciada, eu preciso declarar o construtor, vou criar um construtor aqui, dentro desse construtor eu quero definir o valor inicial da lista.

[05:20] Eu vou ter que acessar o objeto de acesso a dados, “AgendamentoDAO” e eu vou importar a referência, eu vou chamar esse “AgendamentoDAO” de “dao”, vou criar uma nova Instância, só que aqui eu tenho que passar para esse “AgendamentoDAO” uma Instância da conexão com o SQLite e a nossa ViewModel não sabe, ela não tem essa referência para essa conexão.

[05:55] Vamos ter que utilizar aquele nosso serviço do Xamarin Forms de dependência, ela vai criar obter uma instância existente de uma dependência de uma determinada interface, vamos utilizar a interface SQLite para obter o objeto que implementa as funções do SQLite lá do lado do projeto do Android, para obter essa referência o que temos que fazer?

[06:28] Temos que acessar uma “DependencyService”, o serviço de dependência do Xamarin Forms, vamos importar a referência, eu uso um método para pegar uma referência do “ISQLite”, para pegar a referência é o “get” e aqui no generics eu passo a interface, eu passo aqui o abre e fecha parentes e com isso eu também tenho acesso aos objetos dessa interface, ou melhor, aos métodos dessa interface e o método da interface que me interessa, qual é?

[07:13] É a conexão de interface, eu chamo o método “PegarConexao” e nesse método e vou ter o quê? Eu vou ter o retorno que é uma SQLiteConnection, eu vou jogar tudo isso aqui dentro de uma variável que eu vou chamar de “conexao =” a tudo isso aqui, a conexão vai ser passada aqui dentro do “AgendamentoDAO” como parâmetro.

[07:40] Com isso eu já vou conseguir utilizar o objeto de acesso a dados do agendamento, que é o “AgendamentoDAO”, só que o nosso pegar a conexão obtém uma instância nova de uma conexão, ou uma instância já existente, só que essa instância aloca recursos, ela segura arquivos, ela aloca memória, como ele é um objeto que aloca recursos do sistema é sempre uma boa prática a gente fazer o quê?

[08:11] Fazer um Dispose nesse objeto, o que eu vou fazer aqui? Eu posso chamar “conexão.Dispose”, ou melhor, eu vou fazer melhor que isso, eu vou colocar isso dentro de um bloco “using”, ele vai automaticamente chamar o método “Dispose” lá na conexão assim que o processador, assim que o programa passar, ele sair desse escopo da conexão.

[08:45] Agora que eu tenho o objeto de acesso a dados, que o “AgendamentoDAO” eu vou acessar a lista de agendamentos que já foram realizados e que estão lá no banco de dados, então eu faço “dao.” e eu procuro aqui a lista, mas eu não tenho aqui um lista, mas por quê?

[09:04] Porque ainda não implementamos essa lista e é o que vamos fazer agora lá no “AgendamentoDAO”, aqui vamos criar uma nova propriedade que vai armazenar a lista de agendamentos realizados, eu crio uma nova propriedade, “propfull” e eu vou criar como uma lista de agendamentos, eu vou chamar essa lista de “lista” e aqui propriedade pública, porém eu quero que essa propriedade não seja modificada de fora da classe, eu coloco um “private set”.

[09:55] Agora dentro do “get” eu faço o quê? Eu obtenho a partir do banco de dados qual é a lista de agendamentos que já foram realizados, como eu faço isso? Eu tenho que primeiro acessar a conexão e a partir da conexão utilizar algum método, algum mecanismo que permita que e pegue do banco de dados e como o SQLite deixa eu trazer para o código CSharp os objetos que estão armazenados no banco de dados?

[10:31] Eu tenho que chamar um método que vai pegar a tabela de agendamentos, a tabela em inglês é table, “conexão.Table” e aqui dentro do Generics eu passo o tipo da entidade que eu estou trazendo, o tipo do meu modelo, que no caso é agendamento, “Table”, com isso eu já consigo trazer informações da tabela de agendamento, porém o table é um método que traz um tipo chamado “TableQuery” e estamos definindo essa propriedade como sendo uma listagem de agendamento.

[11:12] Eu preciso converter esse table era numa lista de agendamentos então eu faço table e no final do table eu coloco um método para converter isso para uma lista, “.toList” e eu retorno, faço o retorno aqui para ele poder trazer de volta da propriedade.

[11:36] Porém o Visual Studio está reclamando que o tipo agendamento para ele ser utilizado aqui no método “Table” ele tem que ter um Construtor sem parâmetros, temos que fazer o quê? Tem que implementar esse construtor sem parâmetros, vamos lá, onde tem os construtores, eu vou criar mais um construtor, “ctor” “Tab+Tab”, criei um construtor sem parâmetros, vou voltar lá no “AgendamentoDAO”.

[12:] Agora sumiu o erro, temos aqui a lista que está voltando a partir do “AgendamentoDAO, com isso conseguimos acessar o banco de dados do SQLite, agora lá na ViewModel, vamos fazer o quê?

[12:26] Vamos acessar essa lista que acabamos de criar, “dao.lista” vai trazer uma lista de agendamentos, List de agendamento, só que aqui no nosso ViewModel já existe uma propriedade, que um ”ObservableCollection”, que também se chama lista, o que eu quero fazer é transferir todos os agendamentos que estão lá no “AgendamentoDAO”, para nossa ViewModel.

[12:54] Eu vou atribuir uma variável local, “var lista”, que eu vou chamar de “listaDB” porque está vindo do banco de dados, é igual a “dao.Lista” e para cada um dos elementos de listaDB eu quero adicionar esses elementos lá na nossa lista local, a nossa da ViewModel, para cada item DB, dentro de lista DB, eu quero adicionar dentro do lista que está na ViewModel, “this.Lista.Add” para adicionar e eu vou adicionar o “itemDB”.

[13:42] Só que temos que tomar cuidado também de limpar essa lista, antes de iniciar esse laço “foreach”, this.Lista.Clear”, vamos limpar essa lista antes de começar a adicionar nela, nós também precisamos inicializar essa lista, temos que definir aqui o “lista”, esse atributo local como o meu “ObservableCollection” de agendamento, porque sem isso vamos obter erros de referência nula.

[14:14] Agora rodamos a aplicação, vamos ver se esse esquema funciona, agora vou entrar com usuário “joao@alura.com.br”, senha “alura123 e eu vou lá para tela de meus agendamentos, clico no menu, MeusAgendamentos”, agora conseguimos trazer lá do banco de dados aqui para nossa View Model e lá para View, trouxemos aqui os dados de agendamentos já realizados e são os dados que estão gravados no banco de dados.