07

Nova coluna confirmado

Transcrição

[00:00] Quando o nosso usuário faz agendamento do test drive, a aplicação envia para o servidor da Aluracar os dados do agendamento e também salva localmente os dados do agendamento no banco de dados SQLite, por exemplo, eu seleciono aqui um veículo, clico em próximo e vou agendar, vou selecionar primeiro uma data, vou colocar 11 de Fevereiro, vou colocar uma hora, vou colocar 8:15, vou dar "ok" e vou agendar.

[00:34] Quando eu faço isso, eu confirmo o agendamento e ele vai salvar no servidor da Aluracar, mas atenção, agora aconteceu uma falha na hora de agendar o teste drive, ou seja, o servidor da Aluracar, por algum problema, respondeu com a falha, ele não conseguiu gravar os dados lá, o que acontece?

[00:57] Estamos gravando os dados mesmo assim, quando eu clico em "Ok", eu volto para minha lista de veículos disponíveis, só que quando eu entro aqui meus agendamentos, esse agendamento está marcado aqui embaixo, ele está na lista, mas ele não está indicando que esse agendamento falhou, o que precisamos é de uma funcionalidade que permita visualizar quais agendamento falharam, porque?

[01:27] Porque dessa maneira vamos depois tentar enviar novamente esses dados para o servidor da Aluracar, mas o problema é que quando salvamos os dados do agendamento no banco de dados do SQLite, não estamos gravando a informação de que houve falha ao enviar os dados lá para o servidor da Aluracar, não sabemos, depois na nossa lista, quais agendamentos deram falha ou sucesso, o que precisamos fazer?

[01:55] Primeiro armazenar esse valor, que é um verdadeiro ou falso, que vai indicar se houve falha ou não na hora de gravar os dados do agendamento, essa informação está vindo aqui na ViewModel do agendamento, AgendamentoViewModel, temos essa informação no "resposta.IsSuccessStatusCode", que é o resultado do nosso post, a nossa requisição post que foi feito lá na URL do servidor da Aluracar.

[02:30] Primeiro modificar o nosso modelo do agendamento, para poder suportar essa informação, eu tenho que entrar aqui na definição da classe do modelo, que é o agendamento e tenho que definir uma nova propriedade, que eu vou chamar de confirmada, que vai ser um tipo booleano, vai ser verdadeiro ou falso e confirmado vai indicar se houve falha ou sucesso na hora de salvar os dados lá no servidor da Aluracar.

[03:03] Agora eu preciso voltar lá no ViewModel e assim que eu tenho a resposta do sucesso ou falha do agendamento eu preciso armazenar esse valor no objeto agendamento, então eu vou fazer o seguinte, eu faço "this.agendamento.Confirmado = resposta.IsSuccessStatusCode" e eu troco aqui embaixo, onde está "resposta.IsSuccessStatusCode" eu troco por "this.Agendamento.Confirmado".

[03:37] Agora também precisamos exibir a indicação de que o agendamento foi confirmado, como fazemos isso? Temos que ir lá nossa página, que é "AgendamentosUsuarioView.xaml" e colocar aqui alguma indicação visual de que houve a confirmação, de que o agendamento foi confirmado.

[03:59] A maneira mais fácil de fazer isso é colocar um Label, eu vou colocar um Label, colocar um texto e fazer um binding para nova propriedade que é a "Confirmado", então "Confirmado" que é verdadeiro ou falso, vai ser refletido aqui nesse Label.

[04:21] Vamos rodar aplicação e vamos fazer alguns testes, vou entrar com usuário "joao@alura.com.br", senha "alura123", vou entrar, agora vou chegar na lista de veículos e vou agendar vários deles, Fiesta, próximo, agendar, salvar agendamento, confirmei, deu falha aqui, mas ele vai salvar no banco de dados, vamos testar depois o valor.

[04:49] Aqui C3, próximo, agendar, vou colocar dia 12, “Ok”, agora ele vai salvar, vou agendar, confirmei, salvou com sucesso, fui para a lista Inicial, Uno Fire, próximo, agendar, confirmo, “Ok”, ele agendou com sucesso, só mais um então, Sentra, próximo, agendar, confirmo, sim, deu falha ao agendar.

[05:17] Agora vamos lá para página de meus agendamentos e vai ver o resultado, aqui o layout está bem tosco, está bem feio, mas só queremos a informação de que o agendamento foi confirmado ou não.

[05:32] Legal, só que do jeito que está sendo exibido, não está bonito, temos que colocar essa informação de que foi confirmada ou não, de uma outra maneira, o nosso cliente a Aluracar quer que que exibamos nas linhas que falharam o agendamento, eles querem que a gente exiba isso com o texto vermelho, vamos ter que trocar a cor de cada linha do agendamento que falhou.

[05:58[Temos que modificar aqui o nosso ListView, aqui dentro do Template do ListView, temos que modificar a cor da fonte, a cor do texto dessas linhas que são exibidas para cada agendamento, eu tenho aqui dentro do detalhe do ListView, eu tenho três Labels, esses Labels eu vou definir aqui a cor do texto e a cor do texto é “textColor”, eu vou definir como um binding, porque esse TextColor vai variar de acordo com a informação que está vindo do nosso ViewModel.

[06:37] Eu faço Binding para onde? Eu faço binding para propriedade que indica se foi confirmado ou não, que é a propriedade Confirmado, está vendo aqui no Binding, eu faço isso para cada um dos três Labels, eu tenho que colocar nesse aqui também e nesse também, esses três Labels vão ter a cor modificada de acordo com o valor da propriedade confirmado.

[07:06] Mas o confirmado é verdadeiro ou falso, então confirmado não é uma cor, como fazemos para poder traduzir um valor qualquer em um outro tipo? Como já vimos anteriormente, temos algumas classes de conversão de dados, dentro do Xamarin Forms isso é permitido através de uma interface chamado “IValueConverter”.

[07:31] Já temos aqui um conversor para converter o valor de verdadeiro para o falso e o falso do verdadeiro, que é o nosso NegativoConverter, estou abrindo aqui, “NegativoConverter”, convertemos um valor para outro, eu vou criar aqui uma nova classe. Vou criar uma “AgendamentoConfirmadoConverter”, esse vai ser o nosso conversor de booleano para cores.

[08:05] Quando eu faço isso eu tenho que também implementar uma interface que é o “IValueConverter”, vou importar o Namespace, agora eu vou implementar essa interface, implementando a interface eu tenho aqui os dois métodos, no conversor, no método Convert, eu tenho que fazer o quê?

[08:31] Eu tenho que pegar o valor que é um object, eu pego o Value e vou converter esse Value em um Booleano, eu vou colocar aqui a conversão para Boolean e vou armazenar isso aqui numa variável, que eu vou chamar de “Confirmado”, “Confirmado =” a value convertido para boolean.

[08:57] Agora, o que eu vou retornar? Eu tenho que retornar uma cor, eu tenho que retornar a cor do texto que vai ser exibido na listagem, se o Confirmado for verdadeiro, confirmado eu quero fazer o quê?

[09:12] Eu quero retornar a cor preta, “return Color.Black” e se o confirmado for falso eu quero retornar o vermelho, “return Color.Red”.

[09:33] Agora que temos o conversor de booleano para cores, temos que aplicar esse conversor lá no Xaml, vamos lá no “AgendamentosUsuarioView.xaml” e vamos definir aqui qual é o conversor, no Binding para a propriedade TextColor, eu venho aqui no binding Confirmado e adiciono também o Converter dele, o conversor dele, eu vou pegar de onde?

[10:03] Eu tenho que pegar de “StaticResource” e eu tenho que informar a chave desse conversor, mas para eu informar a essa chave eu tenho que ter aqui dentro do meu Xaml informado já qual é o conversor que eu vou usar, não

informamos para essa página, para esse Xaml que existe um conversor de Booleano para cores, eu preciso fazer isso antes de aplicar o conversor aqui no Binding.

[10:37] Vamos criar uma nova Tag “ContentPage” e colocar aqui “Resources”, que são os recursos dessa página e aqui dentro eu preciso colocar o dicionário de recursos, “ResourcesDictionary”, que é o dicionário de recursos e aqui vai a um ou mais conversores, eu posso ter mais de um conversor, mais de um item dentro dessa Tag, eu vou colocar aqui o quê? O conversor de booleano para cores, que é o AgendamentoConfirmadoConverter.

[11:11] Eu vou procurar aqui “AgendamentoConfirmado”, não apareceu nada, por que não apareceu nada? Porque eu tenho que especificar qual é o namespace dessa classe, eu não tenho o Namespace ainda, que é o Namespace da pasta de conversores, como eu defino Namespace em uma página Xaml?

[11:31] Eu faço “xmlns”, ou seja, namespace do XML, dois pontos e vou dar um apelido para esse Namespace, eu vou chamar de “Converters=” e eu vou procurar aqui qual namespace que eu preciso para colocar o conversor, que é o “TesteDrive.Converters”.

[11:58] Agora eu tenho acesso aos conversores através do NamespaceConverter, ou melhor, do prefixo Converters e eu faço dois pontos e olha, tem tanto o “NegativoConverter” que já usamos antes, quando “AgendamentoConfirmadoConverter”, eu vou utilizar esse daqui e eu vou colocar o quê?

[12:17] Eu vou colocar também a chave que é “x:Key” e eu vou chamar esse cara de “ConfirmadoConverter”, essa chave eu tenho que aplicar nos Binding aqui embaixo, eu vou colocar “StaticResource”, que é igual “ConfirmadoConverter”, agora falta colocar esses conversores nos outros Labels também, porque eu tenho aqui mais Binding que estão relacionados à propriedade Confirmado e agora estou colocando os Converters também.

[12:54] Agora vamos rodar aplicação e ver o resultado, vou entrar com usuário “João@alura.com.br”, senha “alura123”, vou entrar, agora vamos lá para o menu e vou selecionar meus agendamentos, agora ficou tudo vermelho, por que isso?

[13:14] Porque lá no começo não tínhamos ainda informação da confirmação ou não do agendamento e nem gravávamos, nem tínhamos essa coluna no banco de dados agendamento do SQLite, por isso quando criamos essa coluna automaticamente, todos os registro que já existiam no banco de dados, a coluna Confirmado assumiu o valor padrão que é falso.

[13:40] Por isso que os agendamentos mais antigos estão vindo em vermelho, a partir do momento que colocamos a coluna confirmado e começou a definir essa coluna a partir do valor que vinha lá do servidor da Aluracar, começamos a ter aqui em preto as linhas dos agendamentos que deram sucesso e em vermelho os agendamentos que falharam.

[14:03] Com isso, conseguimos ver como fazer um conversor de booleano para cores e também como exibir na tela esse conversor, através do binding da propriedade confirmado. Vimos também como criar uma nova coluna no banco SQLite e nem precisamos apagar o banco, olha que interessante, já tínhamos um banco de dados lá e fomos mudando a estrutura desse banco, adicionando uma coluna nova.