09

PostAsync

Transcrição

[00:00] Bom, então, agora a gente vai começar a mudar a aplicação para poder salvar o agendamento para valer. A gente não quer mais simplesmente mostrar uma mensagem de mentirinha para o usuário. Então, a gente vai tirar essa mensagem aqui, a que está informando ele que foi salvo o agendamento.

[00:20] A gente não está salvando nada ainda. Então, a gente vai chamar um método que está lá no viewmodel, do agendamento do test drive. Então, a gente coloca this.ViewModel e a gente vai chamar o método que salva agendamento, eu vou chamar de SalvaAgendamento.

[00:46] Agora, esse método não está implementado ainda, a gente vai ter que criar isso lá no viewmodel. Então, a gente vai na referência do viewmodel, vamos lá para o agendamento viewmodel e a gente vai programar esse novo método lá. Então, aqui embaixo, vamos colocar um public void.

[01:07] E aí, o método SalvaAgendamento. Então, agora a gente vai começar a programar a chama para o serviço, para postar os dados, para ele enviar para o serviço os dados do agendamento. Agora, que dados são esses? E onde que está esse serviço que salva agendamento.

[01:27] Bom, felizmente alguém me mandou aqui essas informações, sem elas, eu não consigo postar isso lá no serviço, eu não sei nem como acessar se eu não tiver essas informações direitinho, tá bom? Então, eu tenho aqui o endereço, http://aluracar.herokuapp.com/salvaragendamento (http://aluracar.herokuapp.com/salvaragendamento).

[01:51] Então, a gente vai ter que chamar essa URL aqui, com o método http post e o conteúdo, vão ser vários campos aqui, que a gente vai passar como string e o conteúdo vai ter que ser num formato string, porém num formato Json. Então a gente vai começar a implementar isso agora.

[02:12] Então, a URL do serviço é https://alugacar.herokuapp.com/salvaragendamento (https://alugacar.herokuapp.com/salvaragendamento). Vou renomear aqui o método para ficar igualzinho. Então, invés de SalvaAgendamento, vou colocar SalvarAgendamento.

[02:44] Bom, essa string não vai ficar solta aqui, ela tem que ir para algum lugar, a gente vai ter que armazenar em algum lugar. Então, a gente vai criar uma constante string, que eu vou chamá-la de URL *POST*AGENDAMENTO. Então, como vai ser utilizada numa chamada post.

[03:04] Então, o URL *POST*AGENDAMENTO. Agora, eu vou mover essa constante lá para cima, lá para o início da classe e aqui eu vou chamar o URL *POST*AGENDAMENTO para salvar os dados no serviço. Bom, como a gente viu antes, existe a biblioteca com a classe http client.

[03:29] Essa classe vai permitir que a gente acesse vários métodos do http, a gente já fez isso com o get string async e a gente vai usar ele para fazer o post também. Então, primeiro, eu vou ter que declarar aqui uma instância da http client. Então, HttpClient, resolvendo aqui a referência, né?

[03:54] Então, eu vou chamar de cliente = new HttpClient. Então, aqui com esse cliente, eu vou chamar o método que corresponde à chamada post. Então, eu coloco: cliente.post e aparece aqui para mim o PostAsync. Com o PostAsync, eu tenho aqui a opção de colocar URL...

[04:23] Eu vou colocar a nossa URL *POST*AGENDAMENTO, em seguida, eu tenho aqui o conteúdo que vai no corpo da requisição http post e esse conteúdo, esse corpo, ele vai ter os dados do agendamento. Agora, só para a gente testar, eu vou colocar o corpo dessa requisição post, eu vou colocar ela como uma string vazia.

[04:52] Isso vai falhar, lógico que não vai funcionar isso, porém, eu vou conseguir fazer uma requisição. Eu quero testar o resultado, apesar do resultado ser uma falha, eu quero pelo menos testar esse retorno, para tomar alguma ação no nosso aplicativo. Então, aqui eu passei a string vazia, mas na verdade...

[05:16] Então, agora a gente vai colocar aqui uma variável que eu vou chamar de conteúdo, ela não existe ainda e vou declarar ela aqui em cima. O conteúdo, ela vai ser uma http content. Então, a gente pode criar esse conteúdo que vai ser o corpo da nossa requisição, que vai conter os dados do agendamento.

[05:37] A gente pode criar ela, essa variável conteúdo, como uma StringContent () e aqui dentro eu vou passar o quê? Eu vou passar uma string Json. Então, aqui eu vou colocar new StringContent () e aqui eu passo um Json. Então, esse Json, eu vou colocar uma string vazia mesmo.

[06:02] E aqui, eu coloco o Encoding. Então, o Encoding vai ser UTF8 e aqui, eu coloco media type, que é o tipo de dado que a gente está passando. A gente está passando uma string, porém essa string vai ser o Json. Então, eu vou colocar aqui aplication/json.

[06:27] Muito bem, então agora a gente já tem uma chamada que pode ser feita no http client, essa chamada vai ser feita lá no serviço como post. Então, agora esse post async, ele tem uma resposta. Então, eu vou chamar aqui, uma variável chamada resposta. Então a resposta vai pegar o resultado do post async.

[06:50] Eu quero verificar se essa resposta, se esse retorno do servidor indicou sucesso ou não, nessa postagem do novo agendamento. Então, eu vou fazer essa chamada, só que olha só, o post async, ele é uma awaitable, ele é uma task awaitable.

[07:09] Então, eu preciso colocar aqui a palavra await, porque eu quero esperar que esse método assíncrono, ele retorne ao resultado, para eu poder continuar a execução do aplicativo. Bom, então o await, ele exige que eu marque o método que tem o await lá dentro...

[07:03] Que eu marque esse método como um assíncrono. Então, eu coloco aqui async, agora que ele está marcado como assíncrono, eu posso obter a resposta para saber se a nossa requisição para o serviço foi bem-sucedida ou não. Então, eu coloco aqui embaixo, if (resposta.IsSuccessStatusCode).

[07:55] Ou seja, se a requisição foi bem sucedida, então eu vou tomar uma decisão, alguma coisa aqui em baixo, senão... Então, else, eu vou tomar outra decisão. Então, o que eu vou colocar aqui? Eu quero simplesmente enviar uma nova mensagem para o sistema de mensageria do Xamarin Forms.

[08:19] Porque como a gente está aqui na viewmodel, eu quero lançar uma mensagem falando: “Olha, essa requisição foi bem sucedida, você conseguiu agendar” ou “falhou a requisição, exiba para o usuário uma mensagem informando que ele não conseguiu postar os dados para o agendamento do test drive.

[08:42] Então, vamos lá. O que eu vou colocar aqui em caso de sucesso é o MessingCenter.Send e aqui eu passo como objeto o tipo agendamento. Então, aqui eu vou passar uma mensagem agendamento e aí, aqui dentro, eu passo o this.Agenamento, que é o objeto que guarda os dados do agendamento.

[09:13] E aqui na mensagem, eu vou colocar aqui um novo, um novo nome de mensagem, que eu vou chamar de SucessoAgendamento. E aí, em caso de falha, eu vou fazer alguma coisa parecida com isso, só que invés de passar o agendamento, eu vou passar aqui uma exceção...

[09:37] Um ArgumentException e aí, no objeto que a gente vai mandar para a mensagem, eu vou chamar de new ArgumentException, então a gente está lançando uma exceção, falando que tem algum erro nos argumentos que estão sendo passados para postagem.

[09:55] Então aqui... e essa mensagem vai ter um nome diferente, ela vai ter que ter um nome... algo como: FalhaAgendamento. Então, FalhaAgendamento vai tratar de quando a postagem não foi bem sucessiva. Então, se o serviço der erro, ele vai lançar uma mensagem como falha agendamento.

[10:20] Agora, não adianta só enviar essas mensagens, se não tiver ninguém para receber, então a gente tem que fazer a outra ponta, tem que fazer alguém assinar essas duas mensagens diferentes, para poder exibir para o usuário o texto, o alerta referente a cada uma das situações.

[10:38] Então, para isso, a gente vai entrar na view, no agendamento view e aqui a gente vai assinar essas duas mensagens. Então, a gente vai aqui no método OnAppearing, então a gente vai colocar MessagingCenter.Subscribe e a gente vai colocar o tipo da mensagem aqui.

[10:58] Primeiro agendamento, para receber o agendamento e aí, passando o this como referência de quem está assinando esse evento, essa mensagem, eu vou colocar aqui o tipo da mensagem, que é aquela SucessoAgendamento. E aí, a gente vai colocar aqui no lambda o método.

[11:28] Então, aqui eu passo o msg e aqui dentro, vai ter o código que vai tratar, que vai ser executado quando a mensagem chegar aqui na view. Então, nesse caso, a gente vai exibir uma mensagem para o usuário com o DisplayAlert. Então, aqui vai ter um título que eu vou chamar de Agendamento.

[11:53] E aqui, a mensagem que vai para ele quando der sucesso na postagem. Então, aqui eu coloco Agendamento salvo com sucesso! E aqui, o que vai no botão da dialog, da mensagem, eu coloco o ok, para ele fechar essa mensagem. Agora, quando acontecer algum erro, a gente tem que assinar uma outra mensagem também.

[12:29] Então MessagingCenter.Subscribe e aí, aqui a gente está recebendo, no caso da falha, um ArgumentException. Então, ArgumentException e aqui dentro, a gente coloca o objeto que vai assinar a mensagem, que é o this e aqui o nome da mensagem que é FalhaAgendamento.

[12:56] E aqui embaixo vai o código que vai tratar a mensagem, o código que vai ser executado quando a mensagem chegar na nossa view. Então aqui, eu coloco um outro DisplayAlert e passo aqui dentro o título Agendamento, aí, eu passo uma mensagem falando que aconteceu alguma coisa muito séria...

[13:19] Alguma coisa errada na aplicação. Então, eu coloco aqui: “Falha ao agendar o test drive! Verifique os dados e tente novamente mais tarde!”, e aí, passo aqui o título que vai no botão, que é o ok. E com isso, eu tenho a assinatura desses dois eventos, dessas duas mensagens de sucesso e de falha.

[13:54] Só que eu também tenho que fazer o quê? Eu tenho que cancelar a assinatura, quando esse formulário for fechado, então aqui em baixo no OnDisappearing, eu coloco MessagingCenter.Unsubscribe e aqui para o sucesso, eu coloco (this, “SucessoAgendamento”).

[14:19] E para o caso da falha, a mesma coisa, MessagingCenter.Unsubscribe e passo aqui dentro a instância (this, “FalhaAgendamento”). Então, essa mensagem também vai ter a assinatura cancelada quando o formulário for fechado.

[14:41] Então, FalhaAgendamento. Então, agora a gente vai com essa parte pronta, a gente vai rodar a aplicação e ver o que acontece. Agora rodando o emulador, vai rodar a aplicação, está rodando o aplicativo agora. Legal, agora eu vou selecionar o veículo, clicar no próximo, preencher os dados, ok?

[15:15] Email... E aí, vou clicar no agendar, confirmo aqui o agendamento. Então, olha só: “Agendamento. Falha ao agendar o test drive! Verifique os dados e tente novamente mais tarde!”. Ok, a nossa mensagem de falha está funcionando. Agora, vamos ver o que aconteceu.

[15:39] Eu vou colocar um breakpoint aqui nessa linha if (resposta.IsSuccessStatusCode). Então, coloco breakpoint aqui e vou clicar no agendar de novo. Quando eu faço isso, ele vai parar aqui nessa linha e o resposta, olha só, resposta está com o StatusCode400, que é o Bad Request.

[16:00] Então, o que está acontecendo? A gente está tentando fazer um post, um método http post para um serviço, só que a gente não passou os dados que ele espera, a gente não passou aqui dentro o Json que é necessário para poder fazer a gravação do agendamento. Então a gente vai fazer isso agora.