03

Interface

Transcrição

[00:00] Agora já instalamos o SQLite, a nossa solução, e toda a manipulação de dados com o SQLite tem que ser feita no projeto Droid, porquê? Porque é um projeto específico para plataforma Android.

[00:16] A implementação da conexão do SQLite varia de um aparelho para o outro, de uma plataforma para outra, o que acontece? Vamos fazer essa manipulação em cima do projeto Droid, do projeto específico para essa plataforma, como quase todo nosso código da nossa aplicação está no projeto Portable e vamos ter que fazer as conexões, o acesso a dados diretamente, vai ter que ser feito dentro do projeto Droid, como vamos fazer para acessar os dados, a partir do projeto Portable para o projeto Droid?

[01:00] Já que, temos uma referência do projeto Portable dentro do projeto Droid, mas não temos uma referência do projeto Droid, no projeto Portable, se adicionarmos uma referência, vou adicionar uma referência do Droid dentro do projeto Portable, vou dar “Ok” e ele vai reclamar que eu não posso adicionar essa referência de projeto dentro do projeto Portable porque isso vai criar uma dependência circular.

[01:33] Uma dependência circular é quando o projeto A referência o B e eu tento fazer o B referenciar o A também, eu não posso criar essa referência circular, como resolvemos isso?

[01:47] Vimos o mesmo problema acontecer quando utilizamos a câmera do Android, que é o recurso nativo do Android, que é a câmera, tentamos utilizar só que não conseguimos fazer essa referência do projeto Android, dentro do projeto Portable, porque isso ia criar também uma referência circular, para resolver isso, como já vimos nas aulas anteriores, temos um componente do Xamarin Forms chamado serviço de dependência ou DependencyService.

[02:20] Ele permite que eu obtenha uma instância de algum objeto, de algum componente que está lá no outro projeto Droid, a partir de uma interface comum, que no nosso caso da última aula era o ICamera, que era uma interface que era comum aos dois projetos e conseguimos utilizar um método do ICamera, que é o TirarFoto.

[02:53] O Xamarin Forms resolveu para gente, descobriu qual é o objeto, qual é o componente do projeto Droid, que podia ser utilizado no projeto Portable, com isso resolvemos o problema de dependência circular.

[03:07] Da mesma maneira precisamos ter uma interface nova que vai poder fazer essa comunicação entre os dois projetos e essa interface nova precisa ter algum método que pegue a conexão do SQLite, com essa conexão vamos conseguir acessar o banco, vamos conseguir acessar os dados do banco SQLite e qual vai ser essa nova interface?

[03:32] Vamos criar aqui dentro do projeto Portable uma pasta nova, que eu vou chamar de “Data”, e vou adicionar uma interface, vou adicionar novo item, vou escolher agora o tipo de arquivo “interface” e eu vou colocar aqui uma interface, vou colocar “ISQLite.cs” e nessa interface eu vou criar um método que vai servir para pegar a conexão do SQLite e trazer aqui para o projeto TestDrive (Portable).

[04:13] Eu vou declarar um método novo, na verdade vai uma função, que eu vou chamar de “PegarConexão”, eu errei o nome da interface, vou trocar, coloquei um “r” a mais no final, deixa eu tirar esse “r”, agora ele vai renomear, agora está certinho.

[04:35] Agora eu tenho que mudar o retorno, não pode ser um void, eu tenho que fazer o “PegarConexão” trazer para esse projeto Portable qual é a conexão da SQLite, qual é o tipo que eu tenho que trazer aqui nessa função?

[04:54] Eu tenho que retornar um “SQLiteConnection”, vou resolver a referência, vou importar aqui o “using”. Agora já temos uma interface que vamos poder utilizar para conectar com o banco de dados SQLite.

[05:14] Agora precisamos ter no projeto Droid uma classe que vai ser o componente que vai implementar nessa nova interface “ISQLite”, como que eu faço isso? Eu crio uma nova classe, simplesmente, aqui dentro do projeto Droid, crio uma nova classe, adicionando a classe e vou chamar essa classe de “SQLite_android”.

[05:47] Cada projeto tem que ter a sua classe SQLite para poder implementar de acordo com a plataforma, se eu tiver por exemplo projeto IOS e Windows Phone eu tenho que ter um SQLite para cada uma dessas plataformas e essa classe precisa implementar a interface ISQLite, eu vou fazer isso agora.

[06:09] Vamos colocar aqui “ISQLite”, agora vou resolver aqui, vou importar a referência, não apareceu aqui, vamos voltar no “ISQLite” e ver o que aconteceu, a interface não está pública, eu preciso colocar um “public” aqui, agora vou resolver, vou importar a referência, agora tudo certo.

[06:32] Agora que eu tenho “ISQLite_android” implementando a interface, eu vou obrigar ele a implementar o método que é o pegar conexão, só que agora o Visual Studio está reclamando da minha interface, vamos ver, o tipo “SQLiteConnection” está definido no assembly que não está referenciado, você precisa adicionar uma referência para o assembly “SQLite-net”.

[06:58] Eu vou tentar instalar agora, acho que ficou faltando a instalação desse projeto Droid, vou colocar aqui projeto default “TestDrive.Droid” e vou rodar novamente o install package “sqlite-net-pcl”, vamos rodar aqui e agora eu vou instalar também no projeto Droid que ficou faltando, agora já instalou no projeto Droid também. Vou fechar aqui, agora vamos fazer essa classe “SQLite_android” implementar a interface “ISQLite”.

[07:33] Vamos clicar com o botão direito, vamos em “Quick Actions” e selecionar a opção implementar interface, quando fazemos isso, o “PegarConexão” é criado como uma função aqui dentro da nossa classe, com essa linha aqui para ele lançar uma exceção que não foi implementada ainda. Agora essa função “PegarConexão” vai ter que retornar uma nova instância de “SQLiteConnection”.

[08:01] Vamos ter que acessar “SQLite.SQLiteConnection” e retornar uma instância nova dessa classe, agora eu faço o retorno dessa nova instância, “return new SQLite.SQLiteConnection”, aqui dentro eu passo um construtor e eu vou ter que passar parâmetros porque ele está falando que não tem um construtor vazio, eu tenho que passar parâmetros para esse construtor e o que eu vou ter que passar aqui dentro?

[08:31] Eu vou ter que passar o nome do arquivo que vai conter o banco de dados e o nome desse arquivo do SQLite tem que ter uma extensão DB3, então vou passar aqui uma String “.db3” e qual vai ser o nome do arquivo?

[08:48] Vamos utilizar o banco SQLite na nossa solução TestDrive para armazenar o agendamento, eu vou chamar esse arquivo de “Agendamento.db3”, agora vamos utilizar esse método “PegarConexão” para obter a conexão do SQLite e poder salvar os dados do agendamento e onde vamos fazer isso?

[09:15] Vamos na classe que atualmente está fazendo o salvamento do agendamento, que está enviando para o servidor da Aluracar quais são os dados do agendamento, vamos para a classe ViewModel, “AgendamentoViewModel”, vamos procurar o método que salva os dados do agendamento, que é o método “SalvarAgendamento”, obtemos a resposta da Aluracar e aqui em seguida vamos implementar o código que vai salvar esses dados localmente no banco de dados, como vamos fazer para pegar a conexão que está lá no projeto do Droid?

[09:54] Como fazemos isso no projeto Portable? Aquele método “PegarConexão” está na interface SQLite, temos aqui “ISQLite”, onde temos o “PegarConexão”, só que como vamos obter a instância do objeto que está implementando essa interface “ISQLite”?

[10:28] Temos que utilizar o serviço de dependência do Xamarin Forms e como é o serviço de dependência? É o DependencyService, fazemos “DependencyService.” e agora vamos obter, vamos pegar uma referência nova ou uma já existente de um componente de uma classe que implementa “ISQLite”, fazemos isso com o método “get”, aqui eu passo qual é o tipo que eu estou pegando, estou pegando uma classe, uma instância que implementa a interface SQLite.

[11:04] Feito isso eu tenho que pegar agora dessa instância, eu tenho que obter o método que é o PegarConexão, quando eu faço isso eu tenho acesso, eu vou ter acesso a uma instância do ISQLite e vou conseguir chamar o método PegarConexão, só que como uma conexão com o banco de dados e aloca recursos do dispositivo, do meu aparelho, recursos como por exemplo memória, arquivos, temos que fazer o quê?

[11:34] Temos que garantir que essa conexão é fechada assim que utilizamos, se não vamos mais utilizar a conexão, nós abrimos, em seguida usamos e em seguida fechamos essa conexão, para isso temos o quê?

[11:49] Temos o método Dispose, porque a conexão, o “SQLiteConnection” é um Disposable, podemos chamar o método Dispose ou simplesmente utilizar um bloco “using”. Usamos o “using” e declaramos uma Instância, vamos pegar a conexão, declaramos aqui, dentro desse bloco “using”, declaramos uma instância de uma conexão, vou chamar de conexão que vai pegar a dependência e vai pegar uma conexão pelo método “PegarConexão”.

[12:29] Em seguida eu vou ter que criar um bloco que vai utilizar essa conexão, aqui dentro eu a conexão como “var”. Aqui dentro eu vou poder utilizar essa conexão para fazer o acesso a dados. Como esse método “SalvarAgendamento” está fazendo mais coisas do que deveria, o que eu vou fazer?

[12:51] Eu vou extrair, eu vou tirar esse código que está tratando da conexão com banco de dados, eu vou extrair e vou colocar em outro método, eu seleciono esse trecho de código, clico com o botão direito e vou em Quick Actions para eu poder extrair o método, eu vou chamar esse novo método de “SalvarAgendamentoDB”, já extrai e aqui eu posso utilizar essa conexão com o banco de dados.

[13:26] Agora já podemos testar essa conexão, o que eu vou fazer? Vou colocar um breakpoint aqui, dentro desse “using” e vamos tentar ver se a conexão é estabelecida com o banco de dados, rodando agora, eu vou entrar aqui com usuário “joao@alura.com.br”, senha “alura123”, vou entrar, vou escolher um veículo, clicar no próximo, vou preencher qualquer coisa aqui e eu vou clicar no agendar, confirma agendamento, agora deu uma exceção de referência nula, porque está acontecendo isso?

[14:09] Como vimos no caso da câmera, esquecemos de fazer o registro, de registrar essa instância que implementa o ISQLite, precisamos fazer esse registro para poder utilizar isso no DependencyService, porque o SQLite_android implementa o ISQLite, mas eu não tenho aqui dentro aquela anotação que fizemos para a classe de câmera, que diz que o “SQLite_android” está implementando uma interface que vai ser utilizada pelo serviço de dependência, pelo DependencyService.

[14:47] Vamos dar uma olhada em como fizemos da outra vez, aqui no MainActivity eu tinha marcado esse assembly como uma dependência, eu tinha marcado esse código como uma dependência para ser utilizada pelo DependencyService, vamos copiar esse código e vamos utilizar de forma similar também no SQLite_android, eu coloco aqui, em vez de colocar MainActivity, eu vou colocar o nome da nossa classe que e SQLite_android.

[15:19] Agora vou ter que importar a referência, já importei a referência, agora vou colocar um breakpoint, para vermos se ele para aqui na nosso método de PegarConexão e vou rodar a aplicação de novo.

[15:34] Eu vou colocar o usuário “joao@alura.com.br”, a senha “alura123”, vamos entrar agora, aqui eu vou escolher um veículo, clicar no próximo e aqui eu vou colocar os dados “joao”, fone alguma coisa, e-mail joao@gmail.com.br. Agora eu vou agendar, vou confirmar o agendamento, sim e ele conseguiu chegar aqui no nosso método PegarConexão que está no projeto Droid, que é essa implementação aqui da interface ISQLite.

[16:22] Agora eu vou prosseguir, vou dar um F5 para rodar, agora deu um erro que ele não consegue abrir o banco de dados “Agendamento.db3”, vamos tentar corrigir isso no próximo vídeo.