04

SQLite Connection

Transcrição

[00:00] Agora temos que resolver esse problema da conexão com o banco SQLite, quando instanciamos o SQLiteConnection temos que passar aqui no construtor um caminho do banco de dados, só que estamos passando somente o nome do banco de dados, o nome do arquivo, mas o que precisamos fazer aqui é passar o caminho, passar o caminho completo com inclusive com os diretórios.

[00:30] Eu vou extrair essa string aqui para uma constante que eu vou chamar de “nomeArquivoDB” e eu vou criar um caminho onde vai ser colocado esse arquivo, vai ter que ser um caminho válido para o dispositivo Android, eu vou colocar “nomeArquivoDB” e vou fazer uma concatenação com o diretório, como eu faço uma concatenação?

[01:03] Eu tenho que utilizar o “path”, que é caminho em inglês e vou combinar o diretório com o nome do arquivo, eu vou resolver, vou importar a referência para o SystemIO e vou utilizar um método de combinar, de concatenar o diretório com o nome do arquivo, agora eu preciso informar qual é o diretório onde eu quero gravar, eu quero gravar esse arquivo de banco de dados no cartão SD do meu emulador, como eu faço isso?

[01:39] Eu tenho que passar aqui o nome do caminho para o cartão SD e como eu não gosto de colocar string no lugar de caminhos eu prefiro utilizar constantes, eu coloco aqui a constante que eu vou obter a partir de “Android.OS.Environment”, eu acesso o “ExternalStorageDirectory” e eu acesso o caminho desse diretório, que é a propriedade “path”.

[02:09] Eu vou ter que armazenar esse resultado dessa concatenação em uma variável que eu vou chamar de “caminhoDB”, “var caminhoDB =” e essa concatenação em seguida, eu vou substituir aqui o nome do arquivo por “caminhoDB”, vou colocar aqui os breakpoints para vemos se realmente agora funciona.

[02:38] Vamos rodar a aplicação, vou entrar com usuário “joao@alura.com.br”, senha “alura123”, vou entrar aqui com veículo, próximo, vou digitar aqui “joao”, qualquer telefone, qualquer e-mail, joao@gmail.com. Vou entrar, vou agendar e vou confirmar o agendamento, quando eu faço isso ele parou aqui no “PegarConexão”, agora ele montou o “caminhoDB”, “Storage/emulated/0/Agendamento.db3”, agora ele vai tentar retornar nessa função uma nova instância de uma conexão SQLite.

[03:30] Conseguiu retornar, conseguiu instanciar uma nova conexão SQLite e ele voltou lá para o nosso projeto Portable, com essa conexão, essa conexão podemos utilizar, pode acessar agora para fazer o nosso acesso ao banco de dados.

[03:48] Agora para começar a fazer o acesso a dados, precisamos ter um local apropriado para isso, vamos fazer o quê? Vamos criar uma nova classe somente para fazer o acesso a dados do agendamento, aqui na pasta “Data” onde já temos a interface ISQLite, vamos criar uma classe para acesso a dados de agendamento, essa classe vai ser o quê?

[04:13] Eu vou chamar ela de “AgendamentoDAO”, Data Access Object ou objeto de acesso a dados, criando essa classe vamos colocar aqui as funcionalidades para poder fazer o acesso ao banco de dados e obter os dados do agendamento e também salvar esses dados.

[04:40] Agora, essa classe “AgendamentoDAO” não vai saber como acessar o banco de dados se não tiver a conexão, não podemos acessar dados sem ter uma conexão aberta e como já estamos criando uma conexão fora dela, lá na nossa ViewModel, precisamos receber essa conexão aqui no “AgendamentoDAO”.

[05:05] Eu preciso criar um construtor que receba essa conexão que foi criada externamente, eu vou criar esse construtor agora, para criar o construtor “ctor”, “Tab + Tab”, pronto criou o construtor automaticamente e aqui dentro do construtor eu quero receber um parâmetro para, eu declaro aqui um parâmetro do tipo SQLiteConnection, vou importar a referência, importei e aqui vou dar o nome desse parâmetro, que vai ser “conexao”.

[05:40] Agora eu tenho que passar, eu tenho que definir uma Instância local da minha classe, a “AgendamentoDAO”, que vai manter, que vai preservar essa conexão durante o tempo de vida do “AgendamentoDAO”, eu vou receber aqui a conexão e vou passar para uma instância local que eu vou chamar de “conexao” também, “this.conexao = conexao”, mas eu tenho que declarar essa variável local.

[06:13] Eu declaro aqui no escopo da classe, agora essa conexão, uma vez ela recebida, passada pelo construtor, ela nunca mais vai ser alterada, ela vai ser imutável, para ela ser imutável eu preciso marcar essa variável aqui, esse campo local como “readonly”, para que ele seja somente leitura, com isso a minha conexão dentro do “AgendamentoDAO” vai ser imutável.

[06:44] Assim que o objeto “AgendamentoDAO” é instanciado precisamos garantir o acesso a tabela desse banco de dados, a tabela chamada “agendamento” e como vamos garantir que essa tabela “agendamento” exista? Porque até agora só vimos a conexão, não vimos a criação da tabela, vamos pegar a conexão e a partir dela vamos criar um novo agendamento, como que fazemos?

[07:10] Pegamos a conexão “this.conexão” e vamos criar a tabela “CreateTable”, passamos aqui nesse método qual é o tipo da tabela que queremos criar, o que eu estou passando dentro do CreateTable é na verdade é um modelo de dados que eu quero para minha tabela que vai ser gerada lá no banco de dados, eu passo aqui dentro o “agendamento”, não o “AgendamentoDAO”, mas o agendamento que eu estou importando a referência, está dento do TestDriveModel, esse método vai ser chamado para criar a tabela.

[07:54] Agora você pode estar se perguntando, mas se já criamos a tabela uma vez e o código passar por aqui de novo, se rodarmos a aplicação de novo e ele passar por aqui, será que não vai ter um conflito? Será que não vai criar a tabela de novo?

[08:09] Se passamos o mouse aqui por cima, na descrição do método, ele diz que executa uma instrução SQL, uma instrução “create table if not exists”, ele só vai criar essa tabela se ele ainda não existir no banco, se lá já existir ele vai ignorar, ele vai passar por aqui e não vai fazer nada.

[08:31] Agora que já temos essa classe “AgendamentoDAO” vamos usar para salvar os dados do agendamento, como fazemos? Vamos lá no “AgendamentoViewModel”, no método que já foi criado “SalvarAgendamentoDB”, vamos utilizar o “AgendamentoDAO” para salvar os dados do agendamento, eu crio uma nova instância do “AgendamentoDAO “ que eu vou chamar de “dao”, “new”, agora vou instanciar o “AgendamentoDAO” passando para esse objeto, no construtor, uma instância da conexão que eu já tinha criado aqui dentro desse método.

[09:10] Eu passo a instância da conexão e aqui agora eu vou chamar o “dao” para poder salvar os dados, “dao.” E aqui eu vou usar o método para salvar o agendamento, mas como estamos vendo não existe um método ainda para salvar agendamento, vamos implementar isso agora.

[09:28] Temos que voltar para a classe “AgendamentoDAO”, e vamos criar um método novo para salvar agendamento, o método eu vou chamar de “Salvar”, aqui dentro eu vou receber o quê?

[09:48] Eu tenho que receber um tipo que é o modelo que é do mesmo tipo do modelo que eu estou utilizando para salvar essa entidade, que é o agendamento mesmo, salvar agendamento e aqui dentro o nome desse parâmetro que vai ser “agendamento”. E aqui dentro vamos salvar no banco dados, mas antes de implementar esse método vamos voltar lá

no ViewModel e chamar esse o método, o método não está acessível, porque ele não está público, eu preciso modificar ele para público, agora vou conseguir acessar.

[10:29] Método “Salvar” e aqui dentro eu crio, vou passar uma nova instância do agendamento, eu passo “new Agendamento” e aqui dentro do construtor, eu queria passar dentro do construtor uma lista de parâmetros desse agendamento, vamos fazer uma pequena refatoração aqui no agendamento para poder receber os parâmetros dentro do construtor.

[10:52] A primeira refatoração que vou fazer nessa classe é modificar os “setters” para eles ficarem privados, porque senão qualquer componente externo pode modificar essas propriedades do agendamento e fica a maior bagunça, eu vou colocar aqui “private set” e vou colocar em todos os outros também, aqui e aqui também.

[11:17] Em seguida eu vou remover a propriedade de veículo e no lugar dela eu vou criar duas novas propriedades, que são o modelo e o preço, vou criar a propriedade com get público, que vai ser o “Modelo” e uma outra que vai ser o preço e o preço vai ser “decimal”.

[11:41] Agora eu vou acrescentar um construtor que vai receber esses parâmetros para poder fazer o preenchimento dos dados, crio aqui um construtor “ctor”, “Tab + Tab”, eu vou passar aqui dentro todas as propriedades. Eu passo o “Nome”, vou colocar minúsculo, vou passar o “Fone”, aqui também minúsculo, vou passar o “e-mail”, minúsculo também, vou passar o “Modelo” e vou passar no final o “Preço” do agendamento, “decimal preco”.

[12:31] Agora eu vou “settar” as propriedades de acordo com os parâmetros do construtor, “this.Nome = nome”, “this.Fone = fone”, “this.Email = email”, “this.Modelo = modelo”, “this.Preco = preco”. Agora quando eu for salvar o agendamento eu vou passar todos esses parâmetros para o construtor, voltando lá, aqui no agendamento, eu tenho que passar o quê?

[13:07] Eu tenho que passar o “nome”, não encontrou o nome porque o nosso método está estático, vou modificar aqui, vou tirar o “static” dele, agora ele consegue acessar o “Nome”, tem que passar o “Fone” também, tem que passar o “Email”, tem que passar o “Modelo”, não encontrou o “Modelo”, vou passar “Veiculo.Nome” e “Veiculo.preco”. Com isso vamos passar para o método “Salvar” uma nova instância do agendamento que tem os dados que queremos salvar no banco de dados.