01

Finalizando o fluxo do App com o banco de dados

Transcrição

Com a lista de contatos integrada ao banco de dados, podemos trabalhar com a **integração do formulário** para que cada item criado seja salvo e suas novas informações sejam apresentadas na lista.

O primeiro passo é modificar seu código no `contact_form.dart`. O trecho que pega os dados dos campos com controladores e cria um novo contato apenas o retorna à lista atualmente; então, precisamos fazer com que salve a informação e a mantenha disponível no banco de dados para a lista.

Para realizar este ajuste, adicionamos o comportamento `save()` antes de `Navigator.pop()` em `Padding()`. Dentro, mandamos um novo contato e executamos o `.then()` para termos certeza de que o item foi salvo neste momento. Em seguida, implementamos em `.then()` uma função com valor inteiro recebido, o qual é um `future:` que retorna o `id` salvo no banco de dados. Depois, executamos a linha de código de `Navigator.pop()` dentro de `.then()`.

Assim não precisamos mais devolver o novo contato, dado que esta informação está no banco para que nossa lista a apresente, e por isso podemos deletar o `newContact` de `.pop()`.

```
Padding(
    padding: const EdgeInsets.only(top: 16.0),
    child: SizedBox(
        width: double.maxFinite,
        child: RaisedButton(
            child: Text('Create'),
            onPressed: () {
                final String name = _nameController.text;
                final int accountNumber =
                    int.tryParse(_accountNumberController.text);
                final Contact newContact = Contact(0, name, accountNumber);
                save(newContact).then((id) => Navigator.pop(context, newContact));
            },
        ), // RaisedButton
    ), // SizedBox
), // Padding
```

Já que estávamos retornando essas informações e imprimindo na lista, acessamos o arquivo `contacts_list.dart` e retiramos também o código de `.then()` presente em `FloatingActionButton()`, sem nos esquecermos de formatar o código corretamente.

Com isso, podemos executar o aplicativo para avaliar o novo comportamento testando a adição de um novo contato.

Se o novo item for apresentado na lista com os demais contatos anteriores, significa que nossa integração entre o formulário e a lista foi bem realizada, e portanto nosso banco de dados está salvando e retornando as informações como esperávamos.

É importante notarmos que mantivemos os dados durante todos os testes feitos. Para finalizar, limpamos o banco para testar o comportamento do aplicativo quando não há nenhuma informação, como acontece em seu **estado inicial de uso**.

Há diversa maneiras de realizar esta limpeza, e aqui usamos uma abordagem bem comum e objetiva; no banco de dados em `app_database.dart`, aplicamos uma configuração diretamente no `openDatabase()` em `CreateDatabase()`. Além disso, se adicionarmos uma vírgula após `1` em `version:`, percebemos a sugestão de outras configurações possíveis, e dentre elas há uma de **regressão de versão** do banco de dados geralmente usada para limpá-lo.

Atualmente estamos na versão `1`, mas há possibilidade de aumentar para `2`, `3` ou mais e ter versões incrementais. Isso é comum quando precisamos fazer alguma atualização no banco de dados, como um novo campo do formulário por exemplo.

Quando temos uma versão mais recente como `5` e queremos voltar para a `4`, este comportamento é chamado **regressão** ou `onDowngrade:`, qual nos permite remover o banco e assim limpá-lo facilmente. Neste, temos uma variável constante chamada `onDatabaseDowngradeDelete:` para implementar esta ação.

Como queremos testar o funcionamento, devemos realizar esta regressão; logo, não adianta executar a aplicação utilizando a versão `1`, pois precisamos atualizar o banco de dados para uma versão mais recente `2` neste caso, e depois voltar para `1` que executa o trecho adicionado.

```
Future<Database> createDatabase() {
    return getDatabasesPath().then((dbPath) {
        final String path = join(dbPath, 'bytebank.db');
        return openDatabase(path, onCreate: (db, version) {
            db.execute('CREATE TABLE contacts('
                'id INTEGER PRIMARY KEY, '
                'name TEXT, '
                'account_number INTEGER)');
        }, version: 2,
        onDowngrade: onDatabaseDowngradeDelete,
        );
    });
}
```

Assim, conseguimos limpar o banco de dados e podemos testar executando o aplicativo no emulador com hot restart. Entramos na lista clicando em "Contacts" do Dashboard para abrir e registrar a segunda versão.

Em seguida, voltamos à versão `1` para executar com hot restart novamente e ver a lista limpa de informações. É uma técnica bastante simples e útil dentro outras possibilidades.

Devemos tomar bastante **cuidado** com a limpeza de banco de dados durante o desenvolvimento, pois pode haver risco de perder informações da usuária ou usuário com a produção e publicação do aplicativo. Por isso, é recomendável comentar ou apagar esta linha de código após os testes.

Com toda a integração pronta, podemos testar a aplicação quantas vezes forem necessárias, adicionando novos contatos por exemplo.

Em seguida, focaremos em como como **melhorar nosso código** em relação a esta primeira implementação.