04

Editando alunos

Transcrição

Conseguimos recuperar os dados de um aluno clicado no formulário. O problema é que quando alteramos os dados desse aluno no formulário e salvamos, a aplicação adiciona um novo aluno, em vez de simplesmente salvar as edições. Isso resultará em nomes duplicados na lista.

Vamos modificar o código para salvar as alterações ao em vez de gerar um novo aluno. Se estamos falando do *checkmark*, estamos falando de um clique no botão do menu. Então, vamos no `FormularioActivity.java`, no método `onOptionsItemSelected`, que fala que um item do menu foi selecionado. Encontramos o seguinte:

```
@Override
public boolean onOptionsItemSelected(MenuItem item) {
    switch (item.getItemId()) {
        case R.id.menu_formulario_ok;
            Aluno aluno = helper.pegaAluno(); AlunoDAO dao = new AlunoDAO(this);
            dao.insere(aluno);
            dao.close();
            Toast.makeText(FormularioActivity.this, "Aluno " + aluno.getNome() + " salvo!",

            finish();
            break;
    }

    return super.onOptionsItemSelected(item);
}
```

Repare que dentro dele fizemos uma verificação para saber qual item do menu foi clicado. Então, falamos que caso clicássemos no botão `menu_formulario_ok`, teríamos um comportamento que pega o aluno do formulário e com esse aluno instanciávamos o `dao` e chamávamos o `insere`. Agora, não é sempre que vamos querer chamar o `insere`, pois, podemos estar simplesmente fazendo uma alteração e não querendo inserir um novo aluno.

Para resolver esse problema podemos fazer outro tipo de verificação. Se o aluno tiver um `id` e se não for nulo, significa que estamos simplesmente alterando esse aluno. Acrescentaremos duas linhas abaixo de `Aluno aluno = helper.pegaAluno()` uma verificação `if (aluno.getId() !=null)`. E com isso dizemos que se é diferente de nulo, deve ser alterado, assim, digitamos `dao.altera(aluno)`. Importante que o `Aluno dao = new AlunoDAO(this)` fique acima da verificação.

Caso o `id` do aluno seja nulo, acrescentamos `else` e embaixo dele copiamos e colamos o `dao.insere(aluno)`. O que acabamos de modificar ficará na linha de baixo do `dao.altera`. Teremos:

```
@Override
public boolean onOptionsItemSelected(MenuItem item) {
    switch (item.getItemId()) {
        case R.id.menu_formulario_ok;
            Aluno aluno = helper.pegaAluno();

            AlunoDAO dao = new AlunoDAO(this);
```

```
                if (aluno.getId() != null)  {
                    dao.altera(aluno);
                } else {
                    dao.insere(aluno);
                }
                dao.close();
                Toast.makeText(FormularioActivity.this, "Aluno " + aluno.getNome() + " salvo!",

                finish();
                break;
            }

            return super.onOptionsItemSelected(item);
        }
```

Só falta implementar o método `altera` . Vamos dar um "Alt+Enter" em cima dele e nosso método será criado no *AlunoDAO.java*, no final da página. Ficaremos com:

```
public void altera(Aluno aluno) {
}
```

Vamos inserir o comportamento desse método. Primeiro, precisamos de um banco de dados que possa ser alterado, assim, acrescentamos o `SQLiteDatabase db = getWritableDatabase()` . Uma vez que temos o `WritableDatabase` , precisamos de um método que faça a atualização, mas fazer isso a mão seria extremamente complicado. Portanto, usaremos o `db.update` e para completá-lo usamos a tabela, `"Alunos"` .

```
    public void altera(Aluno aluno)  {
        SQLiteDatabase db = getWritableDatabase();

        db.update("Alunos");
    }
```

Para seguir completando o `db.update` vamos inserir os dados que usaremos, primeiro, o `ContentValue` . Como teremos que criar o `ContentValue` , na linha de cima digitaremos `ContentValues dados = new ContentValues` . Na linha de baixo disso podemos acrescentar o `dados.put("nome", aluno.getNome())` , mas lembra que já fizemos isso antes? Se você rolar sua barrinha para cima vai ver que já temos tudo isso feito no `insere(Aluno aluno)` . O que faremos é extrair o método.

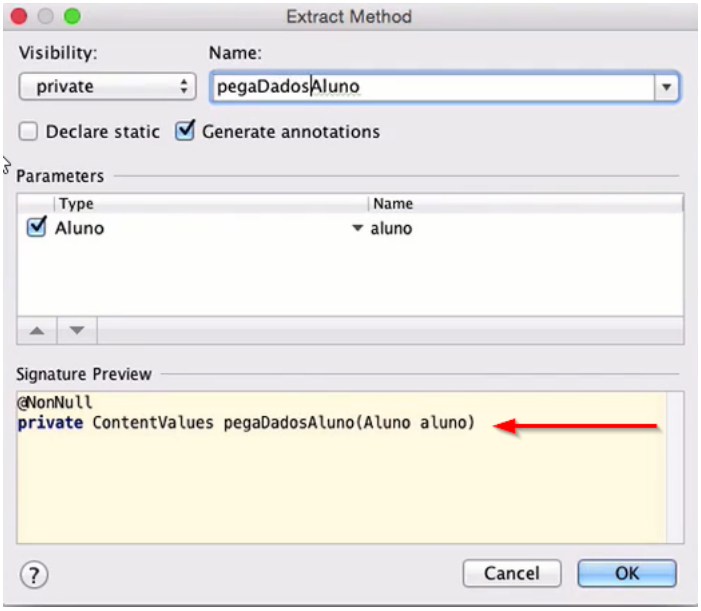
Lembrando que temos o seguinte no `insere(Aluno aluno)` :

```
    public void insere(Aluno aluno) {
        SOLiteDatabase db = getWritableDatabase();

        ContentValues dados = new ContentValues();
        dados.put("nome", aluno.getNome()); dados.put("endereco", aluno.getEndereco());
        dados.put("telefone", a1uno.getTelefone());
        dados.put("site", aluno.getSite());
        dados.put("nota", aluno.getNota());

        db.insert("A1unos", null, dados);
    }
```

Selecionamos com o mouse, no `insere(Aluno aluno)`, o seguinte:

```
ContentValues dados = new ContentValues();
        dados.put("nome", aluno.getNome()); dados.put("endereco", aluno.getEndereco());
        dados.put("telefone", a1uno.getTelefone());
        dados.put("site", aluno.getSite());
        dados.put("nota", aluno.getNota());
```

Clicamos com o botão direito e seguimos por `Refactor> Extract> Method`. Abrirá uma janela pedindo para nomear o método. No campo *Name*, chamaremos de `pegaDadosDoAluno`. Perceba que na *signature preview* conseguimos ver que ele criará um método que recebe um `(Aluno aluno)` e devolve um `ContentValues`.



Damos um "OK" e ele cria o seguinte método:

```
    @NonNull
    private ContentValues pegaDadosDoAluno(Aluno aluno) {
    ContentValues dados = new ContentValues();
    dados.put("nome", aluno.getNome()); dados.put("endereco", aluno.getEndereco());
    dados.put("telefone", aluno.getTe1efone());
    dados.put("site", aluno.getSite());
    dados.put("nota", aluno.getNota()); return dados;
}
```

Agora, vamos voltar no `altera(Aluno aluno)` e completar o `ContentValues` com `aluno`. Teremos `ContentValues dados = pegaDadosDoAluno(aluno)`.

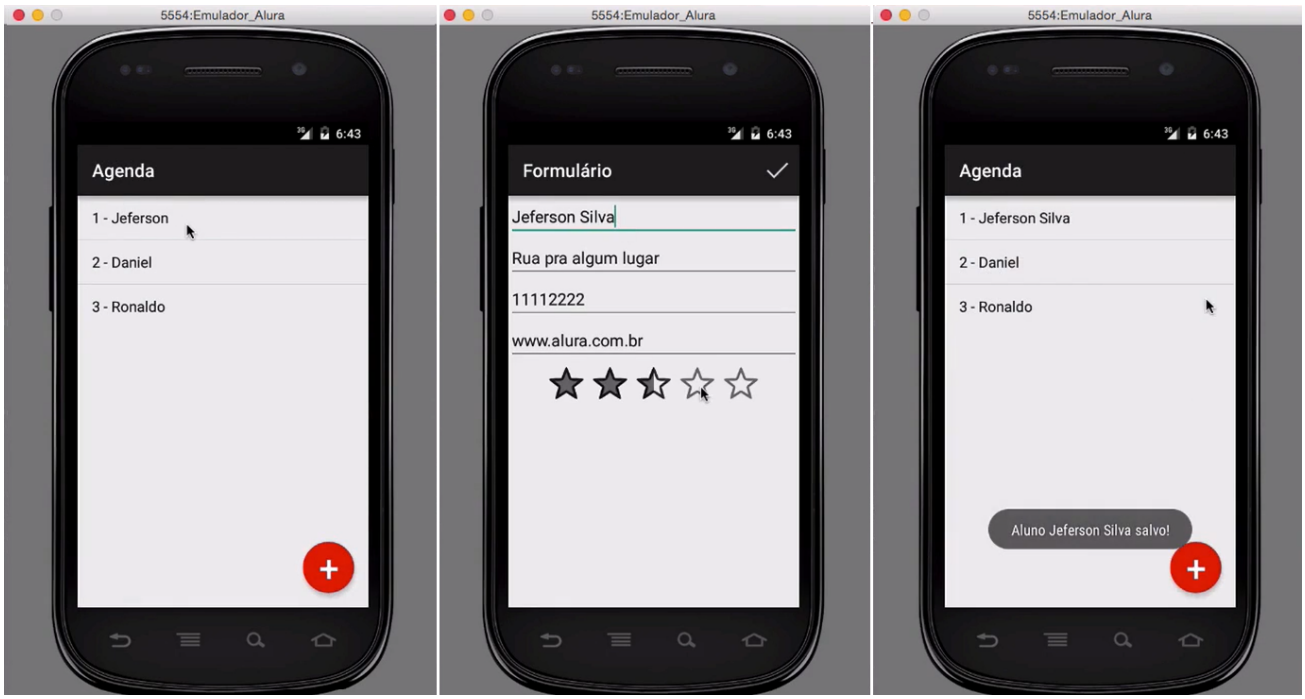
Vamos acessar o `db.update` para completar o restante dos parâmetros do parênteses. Em vez de concatenar o parâmetro, colocaremos um `?` e esse será substituído pelo parâmetro seguinte, que é uma *Array* de *String*, um `params`. Teremos `update("Aluno", dados, "id=?", params)`.

Na linha de cima do `update` vamos construir uma *Array* de *String*, uma `String[] params = {}`. Dentro das chaves colocaremos os valores que entrarão no lugar da interrogação, no caso, o `id` do aluno. Acrescentaremos no parentêses `aluno.getId` e como é um `Long` temos que converte-lo usando a `.toString`. Digitaremos `String[] params = {aluno.getId().toString()}`. Ficaremos com:

```
public void altera(Aluno aluno)  {
    SQLiteDatabase db = getWritableDatabase();

    ContentValues dados = pegaDadosDoAluno(aluno);

    String[] params ={aluno.getId().toString()};
    db.update("Alunos", dados, "id = ?", params);
}
```

Só falta testar! Vamos salvar e dar um *play*.

Vamos editar o aluno "Jeferson", acrescentando nele o sobrenome "Silva" e ao fazer isso constatamos que a aplicação altera o aluno.



Chegamos ao fim! Todos os passos básicos da nossa Agenda estão prontos!