05

Gerando UUID para cada aluno

Atualmente conseguimos realizar uma parte da migration que precisamos fazer para que a nossa App fique totalmente adaptada ao uso de UUIDs. Porém, vimos que os ids dos alunos ainda estão com os valores numéricos e, quando tentamos adicionar novos alunos, os mesmos não vem com ids! Então precisamos resolver os seguintes detalhes:

- Gerar UUIDs para todos os alunos.
- Alterar o método de inserção do banco para que insira com UUIDs.

Dentre passos que precisamos realizar apenas o de geração de UUIDs exige uma atualização, isto é, uma migration do banco de dados em si. Portanto vamos começar por ela!

Buscando todos os alunos do banco de dados

Para isso adicione um novo `case` com valor 3 no `onUpgrade()` do `AlunoDAO`. Em seguida, declare a `String buscaAlunos` com o seguinte valor:

```
"SELECT * FROM Alunos"
```

Agora execute esse script por meio do método `rawQuery()` enviando como parâmetro a `String buscaAlunos` e `null`, pois não enviaremos nenhum argumento para essa *query*. Como vimos em aula, fazemos uso do `rawQuery()` justamente porque queremos o retorno do método, ou seja, atribua o retorno do `rawQuery()` para uma variável do tipo `Cursor`.

Iterando sobre cada registro

Com o `Cursor` precisamos iterar sobre cada registro retornado. Para isso vamos extrair uma parte do código de um método que já faz isso! Conforme vimos em aula, extraímos o código a partir do método `buscaAlunos()`:

```
public List<Aluno> buscaAlunos() {
        String sql = "SELECT * FROM Alunos;";
        SQLiteDatabase db = getReadableDatabase();
        Cursor c = db.rawQuery(sql, null);

        List<Aluno> alunos = new ArrayList<Aluno>();
        while (c.moveToNext()) {
            Aluno aluno = new Aluno();
            aluno.setId(c.getLong(c.getColumnIndex("id")));
            aluno.setNome(c.getString(c.getColumnIndex("nome")));
            aluno.setEndereco(c.getString(c.getColumnIndex("endereco")));
            aluno.setTelefone(c.getString(c.getColumnIndex("telefone")));
            aluno.setSite(c.getString(c.getColumnIndex("site")));
            aluno.setNota(c.getDouble(c.getColumnIndex("nota")));
            aluno.setCaminhoFoto(c.getString(c.getColumnIndex("caminhoFoto")));

            alunos.add(aluno);
        }
        c.close();

        return alunos;
    }
```

Portanto, extraia o método `populaAlunos()`, o código ficará assim:

```
public List<Aluno> buscaAlunos() {
        String sql = "SELECT * FROM Alunos;";
        SQLiteDatabase db = getReadableDatabase();
        Cursor c = db.rawQuery(sql, null);

        List<Aluno> alunos = populaAlunos(c);

        return alunos;
```

```
    }

    @NonNull
    private List<Aluno> populaAlunos(Cursor c) {
        List<Aluno> alunos = new ArrayList<Aluno>();
        while (c.moveToNext()) {
            Aluno aluno = new Aluno();
            aluno.setId(c.getLong(c.getColumnIndex("id")));
            aluno.setNome(c.getString(c.getColumnIndex("nome")));
            aluno.setEndereco(c.getString(c.getColumnIndex("endereco")));
            aluno.setTelefone(c.getString(c.getColumnIndex("telefone")));
            aluno.setSite(c.getString(c.getColumnIndex("site")));
            aluno.setNota(c.getDouble(c.getColumnIndex("nota")));
            aluno.setCaminhoFoto(c.getString(c.getColumnIndex("caminhoFoto")));

            alunos.add(aluno);
        }
        c.close();
        return alunos;
    }
```

Agora com o método extraído basta chamar o `populaAlunos()` enviando o `Cursor` que conseguimos dentro da nossa migration para o `case 3` e devolver a lista de alunos.

Inserindo UUID para cada aluno

Agora precisamos varrer a lista de alunos e para cada um deles precisamos inserir um UUID. Para isso inicialmente vamos criar o script que fará essa atualização no banco de dados. Portanto, crie a `String atualizaIdDoAluno` e adicione o seguinte valor:

```
"UPDATE Alunos SET id=? WHERE id=?"
```

Agora faça um foreach na lista de alunos, e dentro do foreach, faça a chamada do método `execSQL()` enviando como parâmetro a `String atualizaIdDoAluno` e, dessa vez, envie também o segundo parâmetro iniciando por um array de `String`, conforme o exemplo abaixo:

```
db.execSQL(atualizaIdDoAluno, new String[] {});
```

Dentro deste array, envie dois valores: método `geraUUID()` e o `aluno.getId()`. Note que o Android Studio está reclamando do método `geraUUID()`, o motivo é justamente porque não temos esse método ainda! Portanto, crie esse método agora com o atalho **Alt + Enter**. Este método, como o próprio nome diz, ficará responsável em gerar os UUIDs! Portanto, dentro desse método, retorne o seguinte código:

```
UUID.randomUUID().toString();
```

Alterando referências do id de long para String

Veja que o Android Studio não está reclamando mais desse método, porém ele está reclamando da chamada `aluno.getId()`. Em outras palavras, o nosso aluno ainda está atuando com id do tipo `Long`! Agora vamos alterar todas as referências desse id para que atuem como `String`s. Inicialmente vá até a classe `Aluno` e `modifique o tipo do atributo` id `para` String` e seus getter e setter também.

Após fazer essa alteração veja que o Android Studio resolveu aquela linha de código, porém está reclamando de outra, que é justamente no método `populaAlunos()`. Se verificar o erro de compilação verá que o `Cursor` ainda está trabalhando com `Long` para o `id`, portanto altere o código para que atue como `String` usando o método `getString()` ao invés de `getLong()`.

Dentro desta classe não temos mais nenhum detalhe, porém, se irmos no menu **Build** > **Rebuild Project** (obriga o build do gradle completo do projeto) ele mostrará que existem outros pontos do projeto que ainda não estão compilando! O ponto que está dando erro é justamente no método `getItemId()` na classe `AlunosAdapter`, pois esse método devolve apenas ids do tipo `long`. Para resolver esse detalhe basta apenas devolver 0, afinal, não fazemos uso deste método em nossa aplicação.

Testando a aplicação

Antes de testar a App, lembre-se de incrementar mais 1 na versão do SQLite para que o banco seja atualizado! Neste caso deixe a versão com o valor 4. Agora basta apenas testar a aplicação e verificar se todos os alunos estão com UUID assim que chamar a lista de alunos.