Desenvolvimento de Apps para Mobile

Fabrício Tonetto Londero

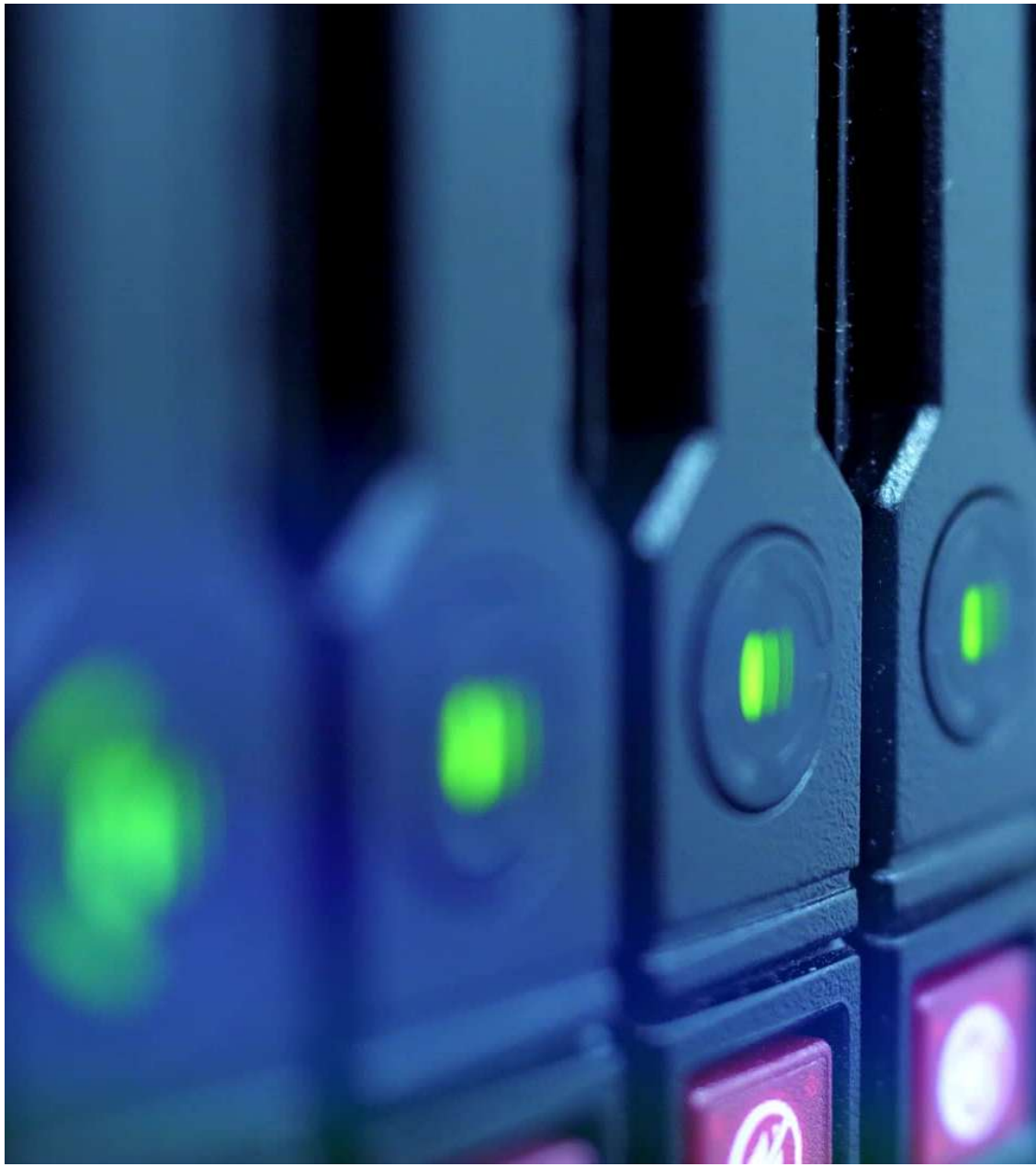
Aula 07 - SQLite

SqLite

- SQLite é um banco de dados relacional gratuito e de código aberto que não depende de um servidor. Ele é uma biblioteca em C que implementa uma base de dados SQL embutida.

SqLite

- Principais características
  - Armazena o banco de dados em um único arquivo
  - Não requer configuração
  - Suporta um grande número de linguagens
  - É leve e simples
  - Funciona sem um servidor dedicado
  - Não exige um administrador
  - Continua funcionando mesmo após quedas de energia



SqLite

Usos

Aplicações mobile, com foco no sistema Android

Dispositivos embarcados e a internet das coisas

Celulares, decodificadores, televisores, consoles de jogos, câmeras, relógios, utensílios de cozinha, termostatos, automóveis, máquinas-ferramentas, aviões, sensores remotos, drones, dispositivos médicos e robôs

Vantagens

Reduz a administração do banco de dados

Permite que novos dados sejam facilmente inseridos em qualquer coluna

Cria uma estrutura mais prática e de fácil uso no dia a dia da empresa

Criador

SQLite foi criado em 2000 por Dwayne Richard Hipp.

Vamos começar criando uma classe

- BancoHelper.java

```
public class BancoHelper extends SQLiteOpenHelper
{

}
```

Atributos

```
private static final String DATABASE_NAME = "meubanco.db";
private static final int DATABASE_VERSION = 1;

// Nome da tabela e colunas – Para o caso de tabela única
private static final String TABLE_NAME = "usuarios";
private static final String COLUMN_ID = "id";
private static final String COLUMN_NOME = "nome";
private static final String COLUMN_EMAIL = "email";
```

Construtor

```
public BancoHelper(Context context)
{
    super(context, DATABASE_NAME, null, DATABASE_VERSION);
}
```

Sobrescrita no OnCreate

```
@Override
public void onCreate(SQLiteDatabase db) {
    String CREATE_TABLE = "CREATE TABLE " + TABLE_NAME + "("
            + COLUMN_ID + " INTEGER PRIMARY KEY AUTOINCREMENT, "
            + COLUMN_NOME + " TEXT, "
            + COLUMN_EMAIL + " TEXT)";
    db.execSQL(CREATE_TABLE);
}
```

Sobrescrita no OnUpgrade

```
@Override
public void onUpgrade(SQLiteDatabase db, int oldVersion, int newVersion) {
    db.execSQL("DROP TABLE IF EXISTS " + TABLE_NAME);
    onCreate(db);
}
```

CRUD - Inserção

```
public long inserirUsuario(String nome, String email)
{
    SQLiteDatabase db = this.getWritableDatabase();
    ContentValues values = new ContentValues();
    values.put(COLUMN_NOME, nome);
    values.put(COLUMN_EMAIL, email);
    return db.insert(TABLE_NAME, null, values);
}
```

CRUD - Consulta

```
public Cursor listarUsuarios()
{
   SQLiteDatabase db = this.getReadableDatabase();
   return db.rawQuery("SELECT * FROM " + TABLE_NAME, null);
}
```

CRUD - Atualização

```
public int atualizarUsuario(int id, String nome, String email)
{
   SQLiteDatabase db = this.getWritableDatabase();
   ContentValues values = new ContentValues();
   values.put(COLUMN_NOME, nome);
   values.put(COLUMN_EMAIL, email);
   return db.update(TABLE_NAME, values, COLUMN_ID + "=?", new String[]{String.valueOf(id)});
}
```

CRUD - Exclusão

```
public int excluirUsuario(int id)
{
   SQLiteDatabase db = this.getWritableDatabase();
   return db.delete(TABLE_NAME, COLUMN_ID + "=?", new String[]{String.valueOf(id)});
}
```

Vamos criar uma Activity

- Acrescente:
  - Um EditText para o Nome do Usuário
  - Um EditText para o Email do Usuário
  - Um botão para “Salvar”
  - Um ListView para a listagem dos usuários cadastrados

Código Java da Activity - Atributos

```
EditText edtNome, edtEmail;
Button btnSalvar;
ListView listViewUsuarios;
BancoHelper databaseHelper;
ArrayAdapter<String> adapter;
ArrayList<String> listaUsuarios;
ArrayList<Integer> listaIds;
```

Código Java da Activity – Método CarregarUsuarios

```
private void carregarUsuarios() {
    Cursor cursor = databaseHelper.listarUsuarios();
    listaUsuarios = new ArrayList<>();
    listaIds = new ArrayList<>();

    if (cursor.moveToFirst()) {
        do {
            int id = cursor.getInt(0);
            String nome = cursor.getString(1);
            String email = cursor.getString(2);

            listaUsuarios.add(id + " - " + nome + " - " + email);
            listaIds.add(id);
        } while (cursor.moveToNext());
    }

    adapter = new ArrayAdapter<>(this, android.R.layout.simple_list_item_1, listaUsuarios);
    listViewUsuarios.setAdapter(adapter);
}
```

Código Java da Activity – OnCreate – Parte 1

```
try
{
    edtNome = findViewById(R.id.edtNome);
    edtEmail = findViewById(R.id.edtEmail);
    btnSalvar = findViewById(R.id.btnSalvar);
    listViewUsuarios = findViewById(R.id.listViewUsuarios);

    databaseHelper = new BancoHelper(this);
```

Código Java da Activity – OnCreate – Parte 2

```
btnSalvar.setOnClickListener(v -> {
    String nome = edtNome.getText().toString();
    String email = edtEmail.getText().toString();

    if (!nome.isEmpty() && !email.isEmpty()) {
        long resultado = databaseHelper.inserirUsuario(nome, email);
        if (resultado != -1) {
            Toast.makeText(this, "Usuário salvo!", Toast.LENGTH_SHORT).show();
            edtNome.setText("");
            edtEmail.setText("");
            carregarUsuarios();
        } else {
            Toast.makeText(this, "Erro ao salvar!", Toast.LENGTH_SHORT).show();
        }
    } else {
        Toast.makeText(this, "Preencha todos os campos!", Toast.LENGTH_SHORT).show();
    }
});
```

Código Java da Activity – OnCreate – Parte 3

```
listViewUsuarios.setOnItemClickListener((parent, view, position, id) -> {
    int userId = listaIds.get(position);
    String nome = listaUsuarios.get(position).split(" - ")[1];
    String email = listaUsuarios.get(position).split(" - ")[2];

    edtNome.setText(nome);
    edtEmail.setText(email);

    btnSalvar.setText("Atualizar");
```

Código Java da Activity – OnCreate – Parte 4

```
btnSalvar.setOnClickListener(v ->
{
    String novoNome = edtNome.getText().toString();
    String novoEmail = edtEmail.getText().toString();

    if (!novoNome.isEmpty() && !novoEmail.isEmpty()) {
        int resultado = databaseHelper.atualizarUsuario(userId, novoNome, novoEmail);
        if (resultado > 0) {
            Toast.makeText(this, "Usuário atualizado!", Toast.LENGTH_SHORT).show();
            carregarUsuarios();
            edtNome.setText("");
            edtEmail.setText("");
            btnSalvar.setText("Salvar");
        } else {
            Toast.makeText(this, "Erro ao atualizar!", Toast.LENGTH_SHORT).show();
        }
    }
});
```

Código Java da Activity – OnCreate – Parte 5

```
listViewUsuarios.setOnItemLongClickListener((adapterView, view1, pos, l) -> {
            int idUsuario = listaIds.get(pos);
            int deletado = databaseHelper.excluirUsuario(idUsuario);
            if (deletado > 0) {
                Toast.makeText(this, "Usuário excluído!", Toast.LENGTH_SHORT).show();
                carregarUsuarios();
            }
            return true;
        });
    });
}
catch (Exception e)
{
    Toast.makeText(this, e.getMessage(), Toast.LENGTH_SHORT).show();
}
```

Exercício

Aplicativo de Lista de Tarefas (To-Do List)
Crie um aplicativo simples de lista de tarefas que deve permitir adicionar, visualizar, atualizar e excluir tarefas. Para isso, crie um banco de dados SQLite chamado **tasks.db** com uma tabela **tasks** contendo os seguintes campos:

- id (INTEGER, chave primária, autoincremento)
- title (TEXT, nome da tarefa)
- description (TEXT, descrição da tarefa)
- status (INTEGER, 0 = pendente, 1 = concluída)

Implementar todas as funcionalidades do CRUD.