



PROGRAMAÇÃO
PARA DISPO-
SITIVOS
MÓVEIS

Ana Karina

Conteúdo da
Aula

Persistência

Exemplo

Programação para Dispositivos Móveis

Aula 14 - Persistência

Ana Karina D. Salina de Oliveira

Faculdade de Computação
Universidade Federal de Mato Grosso do Sul

Programação para Dispositivos Móveis
2018



Roteiro

PROGRAMAÇÃO
PARA DISPO-
SITIVOS
MÓVEIS

Ana Karina

Conteúdo da
Aula

Persistência

Exemplo

- 1 Persistência
- 2 SQLite
- 3 Sintaxe SQL

<https://sqlite.org/docs.html>

<https://developer.android.com/guide/topics/data/data-storage.html?hl=pt-br>



- O Android oferece várias opções para salvar dados de aplicativos de forma persistente.
- A solução escolhida depende das necessidades específicas,
 - se os dados devem ser privados para o aplicativo,
 - se os dados devem ser acessíveis para outros aplicativos (e o usuário),
 - quanto espaço é necessário para os dados, etc.



Como usar armazenamento interno

- É possível salvar arquivos diretamente no armazenamento interno do dispositivo.
- Por padrão, os arquivos salvos no armazenamento interno são privados do aplicativo e não podem ser acessados por outros aplicativo (nem pelo usuário).
- Quando o usuário desinstalar o aplicativo, esses arquivos serão removidos.



Como usar armazenamento interno

- Para criar e gravar um arquivo privado no armazenamento interno:
 - Chame `openFileOutput()` com o nome do arquivo e o modo operacional.
→ Isso retorna um `OutputStream`.
 - Grave no arquivo com `write()`.
 - Feche o stream com `close()`.



Exemplo: Como usar armazenamento interno

```
String FILENAME = "hello_file";
String string = "hello world!";

FileOutputStream fos = openFileOutput(FILENAME, Context.MODE_PRIVATE);
fos.write(string.getBytes());
fos.close();
```



SQLite

- O Android é totalmente compatível com bancos de dados SQLite.
- Todos os bancos de dados criados poderão ser acessados pelo nome em qualquer classe do aplicativo, mas não fora dele.
- O método recomendado para criar um novo banco de dados SQLite é:
 - criar uma subclasse de **SQLiteOpenHelper**
 - e modificar o método **onCreate()**
- possibilitando executar um comando SQLite para criar tabelas no banco de dados.



Exemplo SQLite

```
public class DictionaryOpenHelper extends SQLiteOpenHelper {

    private static final int DATABASE_VERSION = 2;
    private static final String DICTIONARY_TABLE_NAME = "dictionary";
    private static final String DICTIONARY_TABLE_CREATE =
        "CREATE TABLE " + DICTIONARY_TABLE_NAME + " (" +
        KEY_WORD + " TEXT, " +
        KEY_DEFINITION + " TEXT);";

    DictionaryOpenHelper(Context context) {
        super(context, DATABASE_NAME, null, DATABASE_VERSION);
    }

    @Override
    public void onCreate(SQLiteDatabase db) {
        db.execSQL(DICTIONARY_TABLE_CREATE);
    }
}
```



SQLite

- É possível obter uma instância da sua implementação de SQLiteOpenHelper usando o construtor que você definiu.
 - Para gravar no banco de dados chame **getWritableDatabase()**,
 - Para ler no banco de dados chame **getReadableDatabase()**,
- Os dois métodos retornam um objeto SQLiteDatabase que representa o banco de dados e fornece métodos para operações do SQLite.



SQLite

- Você pode executar consultas do SQLite usando os métodos de query() de SQLiteDatabase,
- esses métodos aceitam vários parâmetros de consulta, como
 - tabela a consultar,
 - projeção,
 - seleção,
 - colunas e agrupamento, etc.
- Para consultas complexas, como as que exigem aliases de coluna, use SQLiteQueryBuilder, que fornece vários métodos convenientes para a criação de consultas.



SQLite

- Toda consulta do SQLite retorna um Cursor, que aponta para as linhas encontradas pela consulta.
- O Cursor é sempre o mecanismo usado para navegar pelos resultados de uma consulta de banco de dados e ler linhas e colunas.
- Recomenda-se incluir um campo-chave de valor incrementado automaticamente que seja usado como ID único para encontrar rapidamente um registro.



● Classes de Armazenamento no SQLite:

- **NULL.** Para valores nulos
- **INTEGER.** Valor inteiro armazenado em 1, 2, 3, 4, 6, ou 8 bytes dependendo da magnitude o valor.
- **REAL.** Valor de ponto flutuante, armazenado como 8-bytes.
- **TEXT.** Valor string, armazenado utilizando a codificação (UTF-8, UTF-16BE or UTF-16LE).
- **BLOB.** Valor armazenado exatamente como está.
 - Blobs geralmente são objetos de imagem, audio ou outros objetos multimídia, onde o código binário executável é armazenado como um blob.



Sintaxe da Linguagem SQL: **CREATE TABLE**

- O comando "CREATE TABLE" é usado para criar uma nova tabela em um banco de dados SQLite.
- Atributos a serem especificados da nova tabela:
 - O nome da nova tabela.
 - O banco de dados no qual a nova tabela é criada.
 - O nome de cada coluna na tabela.
 - O tipo declarado de cada coluna na tabela.
 - Um valor ou expressão padrão para cada coluna.
 - Opcionalmente, uma chave PRIMARY para a tabela.
 - Um conjunto de restrições SQL(UNIQUE, NOT NULL, CHECK e FOREIGN KEY).



Sintaxe da Linguagem SQL:

INSERT INTO table VALUES(...);

- Cria uma ou mais novas linhas em uma tabela existente.
- Se a lista de nomes de colunas após o nome da tabela for omitida,
 - o número de valores inseridos em cada linha deverá ser o mesmo que o número de colunas na tabela.
- Se uma lista de nomes de coluna for especificada,
 - o número de valores em cada termo da lista VALUE deve corresponder ao número de colunas especificadas.
- o valor do termo receberá NULL, se nenhum valor padrão for especificado.



Sintaxe da Linguagem SQL:**SELECT**

- A instrução SELECT é usada para consultar o banco de dados.
- Não faz alterações no banco de dados.
- O resultado será uma ou mais linhas de dados.
- Opções para a instrução SELECT:
 - FROM: Determina de onde vem os dados de entrada "TABLE_NAME".
 - WHERE: Filtra os dados de entrada
 - GROUP BY: O conjunto de linhas de resultado é calculado pela agregação dos dados de acordo com a cláusula GROUP BY.
 - DISTINCT: Se for uma consulta "SELECT DISTINCT", as linhas duplicadas serão removidas.



Android

PROGRAMAÇÃO
PARA DISPO-
SITIVOS
MÓVEIS

Ana Karina

Conteúdo da
Aula

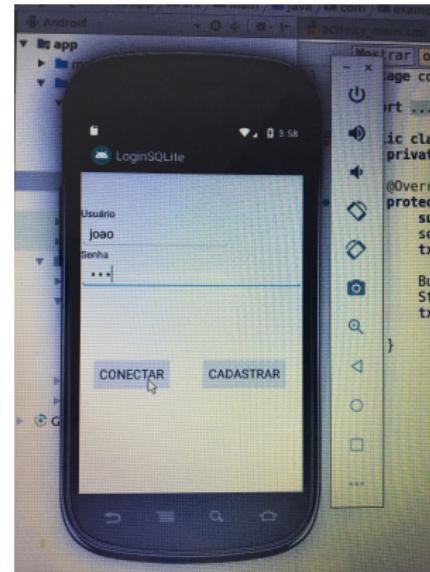
Persistência

Exemplo

Exemplo Login SQLite



Tela inicial





Android

Exemplo Login SQLite

PROGRAMAÇÃO
PARA DISPO-
SITIVOS
MÓVEIS

Ana Karina

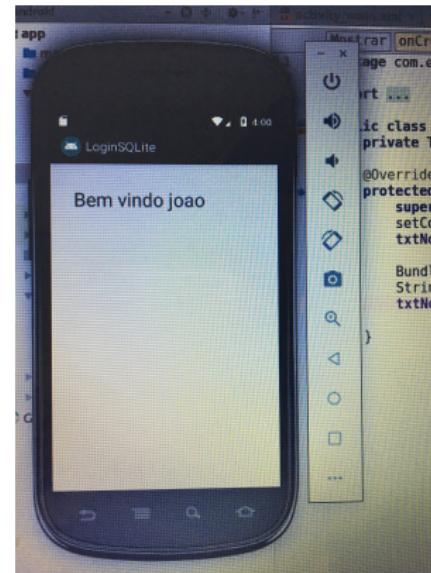
Conteúdo da
Aula

Persistência

Exemplo

Tela Após Login

Activity: Mostrar





Android

Exemplo Login SQLite

PROGRAMAÇÃO
PARA DISPO-
SITIVOS
MÓVEIS

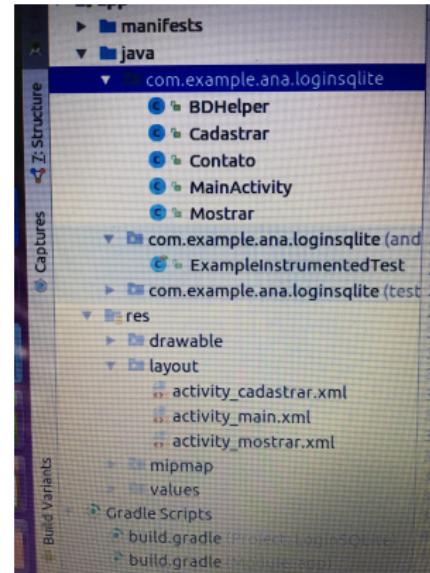
Ana Karina

Conteúdo da
Aula

Persistência

Exemplo

Estrutura dos Arquivos java





Android

Exemplo Login SQLite

PROGRAMAÇÃO
PARA DISPO-
SITIVOS
MÓVEIS

Ana Karina

Conteúdo da
Aula

Persistência

Exemplo

A classe JAVA BDHelper

- Classe para criar o Banco SSqlite
- terá os métodos:
 - public void onCreate(SQLiteDatabase bd)
 - public void insereContato(Contato c)
 - public String buscarSenha(String usuario)
 - public void onUpgrade(SQLiteDatabase bd, int oldVersion, int newVersion)



A classe JAVA BDHelper

Informações para a criação da tabela contato

- private static final int DATABASE_VERSION = 1;
- private static final String DATABASE_NAME = "contato.db";
- private static final String TABLE_NAME = "contato";
- private static final String COLUM_ID = "id";
- private static final String COLUM_NOME = "nome";
- private static final String COLUM_EMAIL = "email";
- private static final String COLUM_USUARIO = "usuario";
- private static final String COLUM_SENHA = "senha";



A classe JAVA Contato

- Classe para acessar e alterar os objetos do Contato
- terá os métodos:
 - public void setNome(String nome)
 - public String getNome()
 - public void setEmail(String email)
 - public String getEmail()
 - public void setUsuario(String usuario)
 - public String getUsuario()
 - public void setSenha(String senha)
 - public String getSenha()