



# Desenvolvimento de Software para Dispositivos Móveis

## Aula 12 - Armazenamento



Professor: Anderson Almada

# Introdução

---

- Armazenamento Privado
- Armazenamento Externo
- Preferências Compartilhadas
- Banco de dados

# Armazenamento privado

- Armazene arquivos privados do aplicativo no sistema de arquivos do dispositivo.
- Para criar um arquivo

```
String fileName = "meuarquivo";
String content = "Hello World";

FileOutputStream outputStream = null;
try {
    outputStream = openFileOutput(fileName,
Context.MODE_PRIVATE);
    outputStream.write(content.getBytes());
    outputStream.close();
} catch (Exception e) {
    e.printStackTrace();
}
```

# Armazenamento privado

Device File Explorer				
Emulator Nexus_5X_API_24 Android 7.0, API 24				
Name	Permi...	Date	Size	
▶ d	lrwxrwxr	1969-12-31 21:0	17 B	
▼ data	drwxrwx-	2019-09-25 10:2	4 KB	
▶ app	drwxrwx-	2019-09-25 10:2	4 KB	
▼ data	drwxrwx-	2019-09-25 10:2	4 KB	
▶ android	drwxrwx-	2019-09-25 10:2	4 KB	
▶ br.ufc.crateus.pratica2	drwxrwx-	2019-09-25 10:2	4 KB	
▼ br.ufc.myapplication	drwxrwx-	2019-09-25 10:2	4 KB	
▶ cache	drwxrwx-	2019-09-29 08:2	4 KB	
▼ files	drwxrwx-	2019-09-29 08:2	4 KB	
meuarquivo	-rw-rw---	2019-09-29 08:2	11 B	
▶ com.android.backupconfir	drwxrwx-	2019-09-25 10:2	4 KB	

# Armazenamento privado

- Para ler um arquivo

```
FileInputStream inputStream = null;
try {
    inputStream = openFileInput(fileName);
    InputStreamReader inputStreamReader = new InputStreamReader(inputStream);
    BufferedReader bufferedReader = new BufferedReader(inputStreamReader);
    StringBuilder sb = new StringBuilder();
    String line;
    while ((line = bufferedReader.readLine()) != null) {
        sb.append(line);
    }
    Log.i("Arquivo", sb.toString());
    inputStreamReader.close();
} catch (Exception e) {
    e.printStackTrace();
}
```

# Armazenamento privado

- Para utilizar o navegador de arquivo do próprio android, pode utilizar de intents.

```
Intent fileintent = new Intent(Intent.ACTION_GET_CONTENT);
fileintent.setType("gagt/sdf");
try {
    startActivityForResult(fileintent, 1);
} catch (ActivityNotFoundException e) {
    Log.e("tag", "No activity can handle picking a file. Showing alternatives.");
}
```

# Armazenamento privado

- onActivityResult

```
@Override
protected void onActivityResult(int requestCode, int resultCode, @Nullable
Intent data) {
    super.onActivityResult(requestCode, resultCode, data);
    if (data == null)
        return;
    switch (requestCode) {
        case 1:
            if (resultCode == RESULT_OK) {
                String filePath = data.getData().getPath();
                File file = new File(filePath);
                Log.i("MainActivity", file.getName());
            }
        }
    }
}
```

# Armazenamento privado

---

- Para utilizar o File como bytes e vice-versa:

```
// Ler um arquivo  
byte[] decodedBytes = FileUtils.readFileToByteArray(file);  
  
// Gravar dados em um arquivo  
FileUtils.writeByteArrayToFile(file, decodedBytes);
```



## Armazenamento externo

---

- Armazena arquivos no sistema de arquivos externo compartilhado. Isso geralmente funciona para arquivos compartilhados entre usuários, como fotos.
- Para operar com o armazenamento externo é necessário permissões:
  - `<uses-permission android:name="android.permission.READ_EXTERNAL_STORAGE"></uses-permission>`
  - `<uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE"></uses-permission>`

# Armazenamento externo

---

- Permissão em tempo de execução (versões mais recentes  $\geq 6.0$ )
- Importe no build.gradle (Module: app)

```
dependencies {  
    ...  
    implementation 'com.karumi:dexter:6.0.0'  
}
```

- Clique em **Sync now** para sincronização do projeto com as dependências

# Armazenamento externo

- onPermissionsChecked, local do código que precisa de permissão

```
Dexter.withActivity(this).withPermissions(  
    Manifest.permission.READ_EXTERNAL_STORAGE,  
    Manifest.permission.WRITE_EXTERNAL_STORAGE  
).withListener(new MultiplePermissionsListener() {  
    @Override  
    public void onPermissionsChecked(MultiplePermissionsReport report) {  
        //Código  
    }  
  
    @Override  
    public void onPermissionRationaleShouldBeShown(List<PermissionRequest>  
permissions, PermissionToken token) {  
  
    }  
}).check();
```

## Armazenamento externo

---

- Antes de operar com o armazenamento externo, é necessário verificar se o armazenamento está montado
- `Environment.MEDIA_MOUNTED`

## Armazenamento externo

```
public boolean isExternalStorageWritable() {
    String state = Environment.getExternalStorageState();
    if (Environment.MEDIA_MOUNTED.equals(state)) {
        return true;
    }
    return false;
}

public boolean isExternalStorageReadable() {
    String state = Environment.getExternalStorageState();
    if (Environment.MEDIA_MOUNTED.equals(state) ||
        Environment.MEDIA_MOUNTED_READ_ONLY.equals(state)) {
        return true;
    }
    return false;
}
```

## Armazenamento externo

---

- O próximo exemplo cria um diretório chamado **Download**
- Dentro desse diretório, cria um arquivo chamado **myData.txt**
- Escreve um conteúdo qualquer nele.

## Armazenamento externo

```
private void writeToSdFile() {  
    File root = android.os.Environment.getExternalStorageDirectory();  
    File dir = new File(root.getAbsolutePath() + "/Download");  
    dir.mkdirs();  
    File file = new File(dir, "myData.txt");  
  
    try {  
        FileOutputStream f = new FileOutputStream(file);  
        PrintWriter pw = new PrintWriter(f);  
        pw.println("Hi , How are you");  
        pw.flush();  
        pw.close();  
        f.close();  
    } catch (Exception e) {  
        e.printStackTrace();  
    }  
}
```

## Armazenamento externo

- Para chamar esse método, vamos chamar no onCreate, dentro do Dexter, mais especificamente no onPermissionsChecked

```
@Override
public void onPermissionsChecked(MultiplePermissionsReport report) {
    if (isExternalStorageWritable()) {
        writeToSDFFile();
    }
}
```



# Preferências compartilhadas

- Armazene dados primitivos privados em pares de chave-valor.

```
SharedPreferences sharedPreferences;
```

```
@Override
```

```
protected void onCreate(Bundle savedInstanceState) {
```

```
    super.onCreate(savedInstanceState);
```

```
    setContentView(R.layout. activity_main);
```

```
    sharedPreferences = getSharedPreferences( "app", Context.MODE_PRIVATE);
```

# Preferências compartilhadas

---

```
public void savePreference(String key, String value) {  
    SharedPreferences.Editor editor = sharedPreferences.edit();  
    editor.putString(key, value);  
    editor.apply();  
}
```

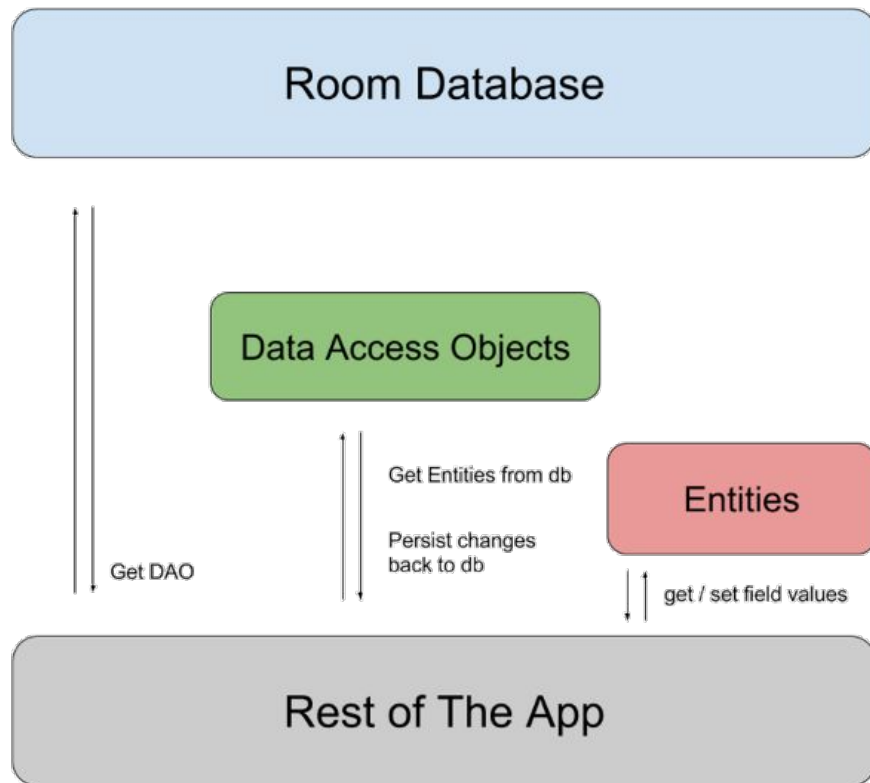
```
public String readPreference(String key) {  
    return sharedPreferences.getString(key, "");  
}
```

# Banco de dados

---

- Por padrão o Android usa o SQLite
- Para facilitar as manipulações no banco de dados, vamos utilizar a biblioteca Room

# Banco de dados



# Banco de dados

---

Annotations	Purpose
@Entity	Creates a SQLite table in the database using a data model class.
@Dao	Create a Data Access Object in the database using an interface class.
@Database	A class with this annotation will create an abstraction for the Data Access Object.
@PrimaryKey	A variable with this annotation will set a primary key for the table.
@Insert	Inserts parameters into the table.
@Update	Updates parameters of an existing table.
@Delete	Deletes parameters of an existing table
@Query	Running SQL query method within the table
@Ignore	Ignores the parameter form the Room database

# Banco de dados

---

- Importe no build.gradle (Module: app)

```
dependencies {  
    ...  
    implementation "androidx.room:room-runtime:2.2.0-rc01"  
    annotationProcessor "androidx.room:room-compiler:2.2.0-rc01"  
}
```

- Clique em **Sync now** para sincronização do projeto com as dependências

# Banco de dados

---

- Primeiro crie o Model

- User

```
int uid;
```

```
String firstName;
```

```
String lastName;
```

- Em seguida, insira as anotações

# Banco de dados

---

```
@Entity
public class User {
    @PrimaryKey(autoGenerate = true)
    public int uid;
    @ColumnInfo(name = "first_name")
    public String firstName;
    @ColumnInfo(name = "last_name")
    public String lastName;
```



# Banco de dados

---

- Model - Código
  - <https://pastebin.com/utnkcEZB>

# Banco de dados

---

- Agora crie a interface DAO de gerenciamento dos dados

```
@Dao  
  
public interface UserDAO {  
    @Query("SELECT * FROM user")  
    List<User> getAll();  
  
    //outros métodos  
}
```

# Banco de dados

---

```
@Query("SELECT * FROM user WHERE uid IN (:userIds)")
```

```
List<User> loadAllByIds(int[] userIds);
```

```
@Query("SELECT * FROM user WHERE uid = :userId")
```

```
User getById(int userId);
```

```
@Query("SELECT * FROM user WHERE first_name LIKE :first AND " + "last_name  
LIKE :last LIMIT 1")
```

```
User findByName(String first, String last);
```

# Banco de dados

---

@Insert

```
void insertAll(User... users);
```

@Insert

```
void insert(User user);
```

@Delete

```
void delete(User user);
```

@Update

```
void update(User user);
```

# Banco de dados

---

- DAO - Código
  - <https://pastebin.com/Aqw9BW6T>

# Banco de dados

---

- Agora crie a classe que é o banco de dados que armazena todos os DAOs

```
@Database(entities = {User.class}, version = 1)
public abstract class AppDatabase extends RoomDatabase {
    public abstract UserDao userDao();
}
```

# Banco de dados

---

- Agora crie uma classe para ser o singleton de acesso ao banco de dados. O nome da classe é **DatabaseClient**
- Código
  - <https://pastebin.com/XRqr3gXp>

# Banco de dados

---

- Agora na activity de acesso ao banco de dados, basta criar a instância e chamar os métodos

```
UserDAO users;
```

```
@Override
```

```
protected void onCreate(Bundle savedInstanceState) {  
    super.onCreate(savedInstanceState);  
    setContentView(R.layout. activity_main);  
    users = DatabaseClient.getInstance(getApplicationContext()).userDao();  
}
```



# Banco de dados

---

- Agora na activity de acesso ao banco de dados, basta criar a instância e chamar os métodos

```
UserDAO users;
```

```
@Override
```

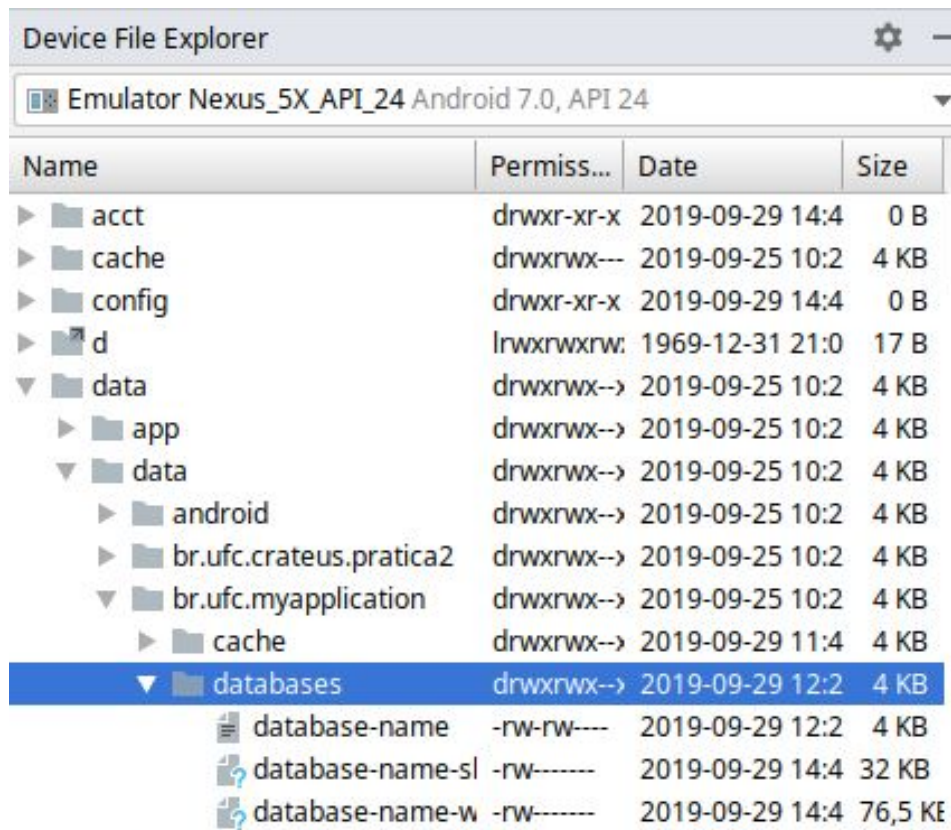
```
protected void onCreate(Bundle savedInstanceState) {  
    super.onCreate(savedInstanceState);  
    setContentView(R.layout. activity_main);  
    users = DatabaseClient.getInstance(getApplicationContext()).userDao();  
}
```

# Banco de dados

---

- MainActivity - Código
  - <https://pastebin.com/dxhPHFBk>

# Banco de dados



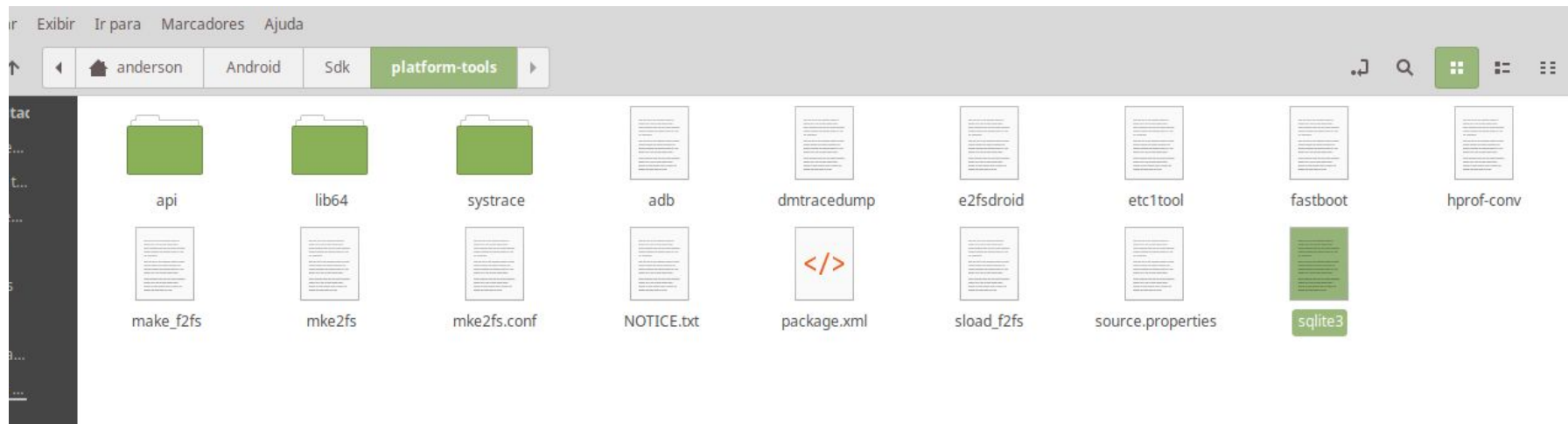
Device File Explorer

Emulator Nexus\_5X\_API\_24 Android 7.0, API 24

Name	Permiss...	Date	Size
▶ folder acct	drwxr-xr-x	2019-09-29 14:4	0 B
▶ folder cache	drwxrwx---	2019-09-25 10:2	4 KB
▶ folder config	drwxr-xr-x	2019-09-29 14:4	0 B
▶ file d	lrwxrwxrwx	1969-12-31 21:0	17 B
▼ folder data	drwxrwx-->	2019-09-25 10:2	4 KB
▶ folder app	drwxrwx-->	2019-09-25 10:2	4 KB
▼ folder data	drwxrwx-->	2019-09-25 10:2	4 KB
▶ folder android	drwxrwx-->	2019-09-25 10:2	4 KB
▶ folder br.ufc.crateus.pratica2	drwxrwx-->	2019-09-25 10:2	4 KB
▼ folder br.ufc.myapplication	drwxrwx-->	2019-09-25 10:2	4 KB
▶ folder cache	drwxrwx-->	2019-09-29 11:4	4 KB
▼ folder databases	drwxrwx-->	2019-09-29 12:2	4 KB
file database-name	-rw-rw----	2019-09-29 12:2	4 KB
file database-name-sl	-rw-----	2019-09-29 14:4	32 KB
file database-name-w	-rw-----	2019-09-29 14:4	76,5 KE

# Banco de dados

- Para gerenciar o banco de dados na linha de comando, pode usar o sqlite3 (Ele se encontra na mesma pasta do adb)
- Caminho: ~/Android/Sdk/platform-tools



# Banco de dados

```

Arquivo  Editar  Ver  Pesquisar  Terminal  Ajuda
anderson@anderson:~$ sqlite3
SQLite version 3.22.0 2018-01-22 18:45:57
Enter ".help" for usage hints.
Connected to a transient in-memory database.
Use ".open FILENAME" to reopen on a persistent database.
sqlite> .open database-name
sqlite> .tables
User                android_metadata    room_master table
sqlite> select * from User;
1|Anderson|Almada
2|Anderson|Almada
3|Anderson|Almada
4|Anderson|Almada
sqlite>

```

Device File Explorer

Emulator Nexus\_5X\_API\_24 Android 7.0 API 24

Name	Permissions	Date	Size
data	drwxr-xr-x	2019-09-29 14:4	0 B
cache	drwxrwx--	2019-09-25 10:2	4 KB
config	drwxr-xr-x	2019-09-29 14:4	0 B
d	lrwxrwxrwx	1969-12-31 21:0	17 B
data	drwxrwx--	2019-09-25 10:2	4 KB
app	drwxrwx--	2019-09-25 10:2	4 KB
data	drwxrwx--	2019-09-25 10:2	4 KB
android	drwxrwx--	2019-09-25 10:2	4 KB
br.ufc.crateus.pratica2	drwxrwx--	2019-09-25 10:2	4 KB
br.ufc.myapplication	drwxrwx--	2019-09-25 10:2	4 KB
cache	drwxrwx--	2019-09-29 11:4	4 KB
database	drwxrwx--	2019-09-29 12:2	4 KB
database-name	-rw-rw---	2019-09-29 12:2	4 KB
database-name.sl	-rw-----	2019-09-29 14:4	32 KB

# Link importante

---

- <https://github.com/Karumi/Dexter>
- <https://developer.android.com/guide/topics/data/data-storage>
- <https://developer.android.com/training/data-storage/room>



# Dúvidas??

E-mail: [almada@crateus.ufc.br](mailto:almada@crateus.ufc.br)