Android-Workshop

Appsfactory GmbH - Lars Lokaizyk - Lead-Developer Android

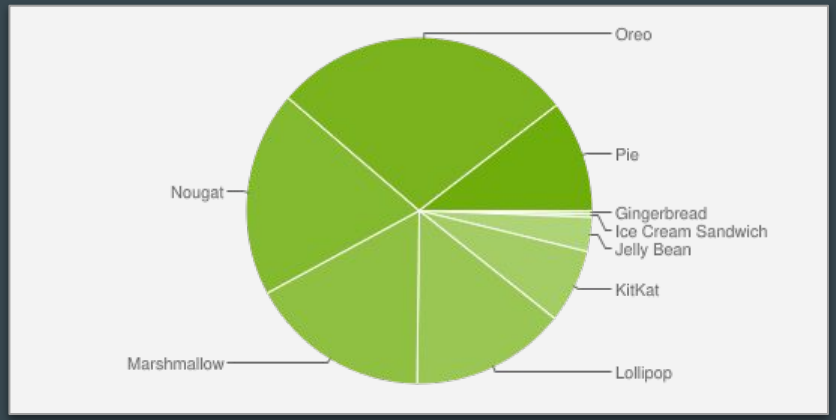
Appsfactory

- Mobil-Agentur für Individual-Apps
- 4 Standorte in Deutschland:
  Leipzig, Hamburg, Erfurt, München
- Was wir machen:
  - Fullstack
  - Android
  - iOS
  - QA
  - Design

- Über 500 veröffentlichte Apps
- Kunden
  - Daimler
  - NDR (Tagesschau-App)
  - Porsche
  - DB
- Was wir suchen:
  https://appsfactory.de/de/jobs/

Grundlagen - Android SDK - Infos

- Linux basiertes System
- Versionen offiziell nach Süßigkeiten benannt (aktuell Android Pie - 8.0)
- Versionen intern als API/SDK level bezeichnet (Android Pie = API level 28)
- Apps legen Mindestlevel fest
- Programmiersprache Kotlin (früher Java)



Version	Codename	API	Distribution
2.3.3 - 2.3.7	Gingerbread	10	0.3%
4.0.3 - 4.0.4	Ice Cream Sandwich	15	0.3%
4.1.x	Jelly Bean	16	1.2%
4.2.x		17	1.5%
4.3		18	0.5%
4.4	KitKat	19	6.9%
5.0	Lollipop	21	3.0%
5.1		22	11.5%
6.0	Marshmallow	23	16.9%
7.0	Nougat	24	11.4%
7.1		25	7.8%
8.0	Oreo	26	12.9%
8.1		27	15.4%
9	Pie	28	10.4%

Grundlagen - Android SDK - Entwicklung

Android SDK

- SDK tools:
  - Emulator
  - Manager
  - ProGuard
- Build tools
  - AAPT
  - dx
- Platform tools
  - ADB
  - SQLite

Android Studio

- basiert auf IntelliJ IDE
- optimiert auf Android-Entwicklung
- beinhaltet Android SDK Tools
- Features
  - Code editor
  - Layout editor
  - Profiler
  - Emulator

Grundlagen - Android SDK - Buildsystem

Gradle

- ähnlich zu Maven, Skriptsprache ist Groovy
- Kompilierung der Dateien (Java/Kotlin) in Byte-Code
- Auflösung von internen Abhängigkeiten
- Signieren von Apps
- Wichtige Dateien
  - settings.gradle -> enthaltene Module
  - gradlew[.sh|.bat] -> Aufrühren von Befehlen
  - build.gradle -> Groovy script (root verzeichnis und für jedes Modul)
  - gradle-wrapper.properties -> Gradle-Version

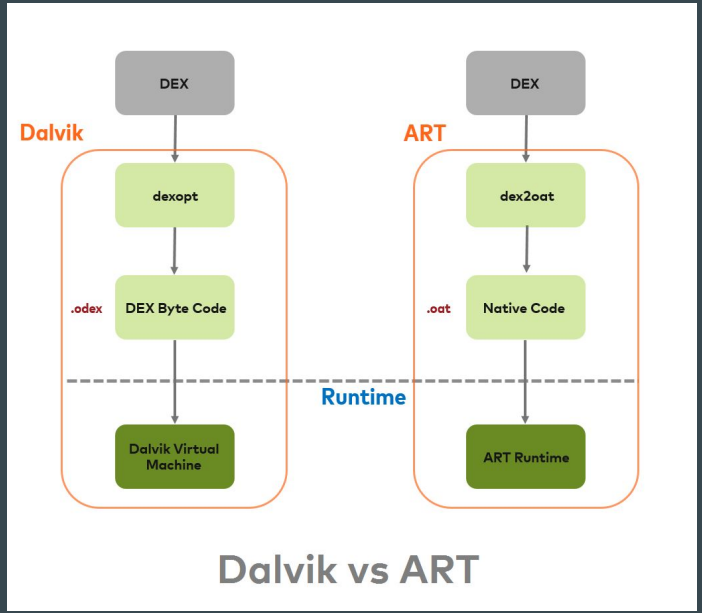
Grundlagen - Android Anwendungen

APK

- Installationsdatei einer Android-App
- ähnlich einer JAR-Datei für Java
- ist komprimiertes Archiv:
  - Bytecode (Dex-Dateien)
  - Ressourcen und Assets

Android Runtime (ART/Dalvik)

- Laufzeitumgebung auf Mobiltelefonen
  -> spezielle JVM
- Verarbeitung der Dex-Dateien
- ART nutzt AOT (Ahead-of-Time) decodierung



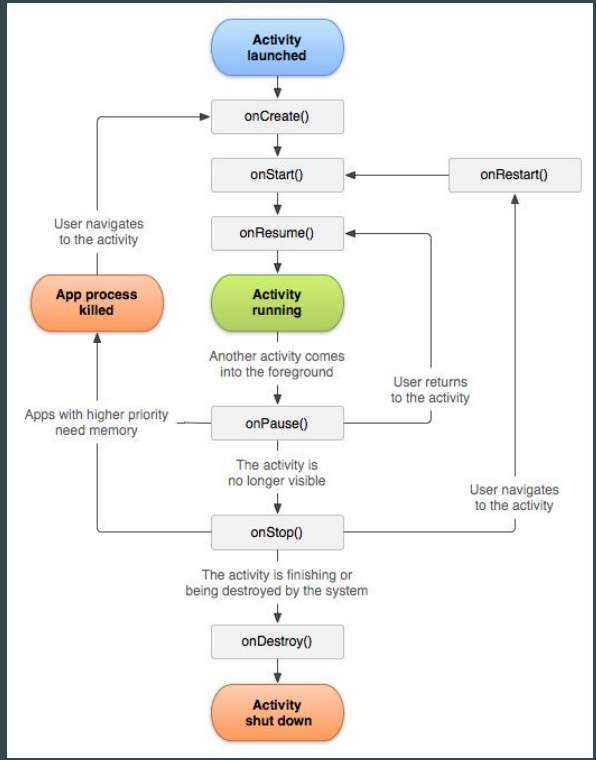
Grundlagen - Komponenten

Activity

- Komponente mit der der User interagiert
- Repräsentiert einen Bildschirm
- Besitzt eine UI

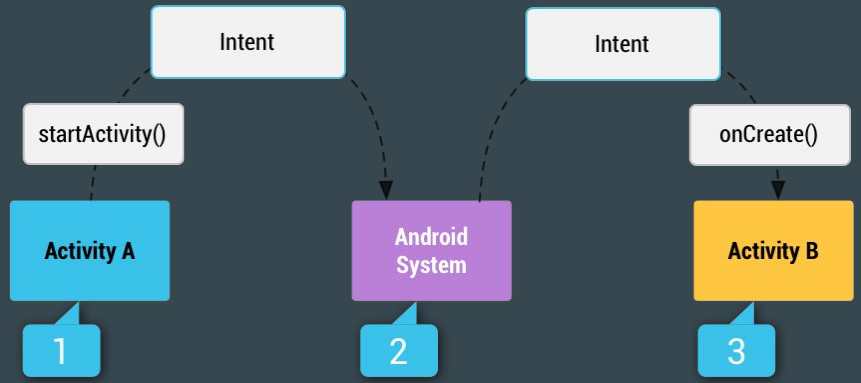
Weitere Hauptkomponenten

- Service:
  läuft im Hintergrund (z.B. Audioplayer)
- BroadcastReceiver:
  Events zur Kommunikation zwischen Komponenten
- ContentProvider:
  Stellt anderen Apps Inhalte zur Verfügung



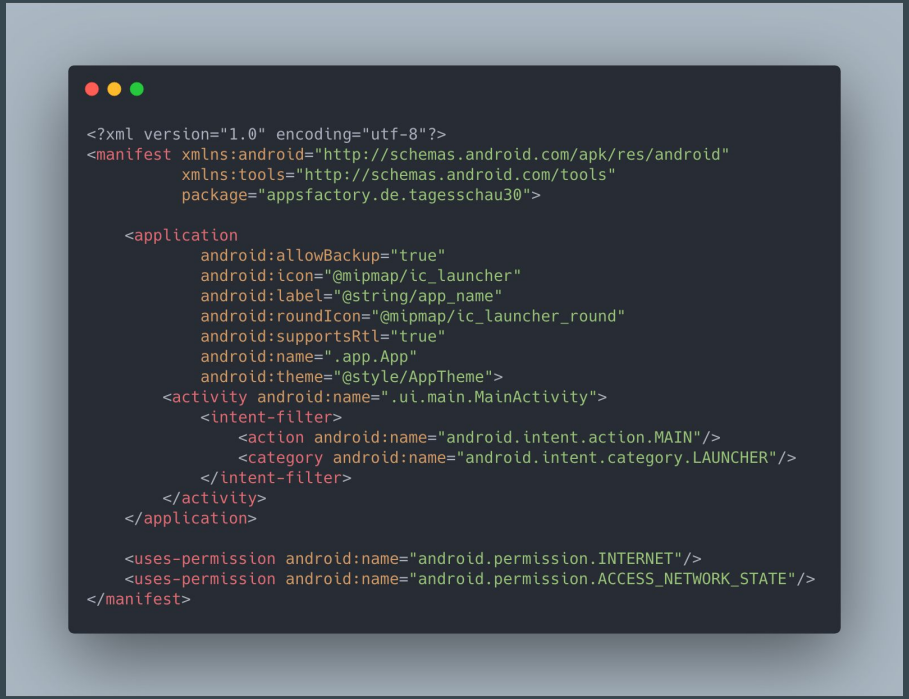
Grundlagen - Intents

- Kommunikation mit Komponenten
  - Starten von Activity/Service
  - Senden eines BroadCast
  - Intent beinhaltet “Action” und “Data”
- können Filter und Kategorien haben
- Daten werden in Bundle gepackt
- Es gibt explizite und implizite Intents
  - explizit: angesprochene App wird direkt angegeben
    -> z.B. Start einer Activity
  - implizit: nur action wird angegeben
    -> kann von mehreren Apps verarbeitet werden, z.B. teilen von Texten (Intent.ACTION_SEND)



Grundlagen - AndroidManifest

- Info über Komponenten der App (Activity, Service, ContentProvider, BroadcastReceiver)
- Erforderliche Berechtigungen (z.B. Internet, Kamera)
- Hard- und Softwareanforderungen



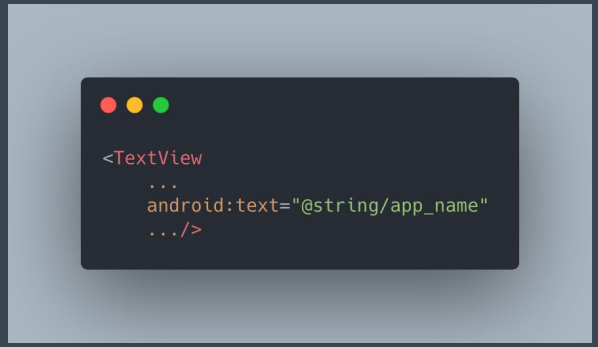
```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
          xmlns:tools="http://schemas.android.com/tools"
          package="appsfactory.de.tagesschau30">

    <application
            android:allowBackup="true"
            android:icon="@mipmap/ic_launcher"
            android:label="@string/app_name"
            android:roundIcon="@mipmap/ic_launcher_round"
            android:supportsRtl="true"
            android:name=".app.App"
            android:theme="@style/AppTheme">
        <activity android:name=".ui.main.MainActivity">
            <intent-filter>
                <action android:name="android.intent.action.MAIN"/>
                <category android:name="android.intent.category.LAUNCHER"/>
            </intent-filter>
        </activity>
    </application>

    <uses-permission android:name="android.permission.INTERNET"/>
    <uses-permission android:name="android.permission.ACCESS_NETWORK_STATE"/>
</manifest>
```

Grundlagen - Ressourcen

- Ressourcen werden im Verzeichnis res gespeichert
  - Colors,
  - Strings (lokalisiert),
  - Drawables,
  - Dimens
  - Layouts,
  - Styles,und weitere
- aus Ressourcen wird R-Datei erzeugt mit eindeutiger ID für jede Ressource:
  - R.[typ].[name] -> z.B. R.string.app_name
  - @[typ]/[name] -> z.B. @string/app_name



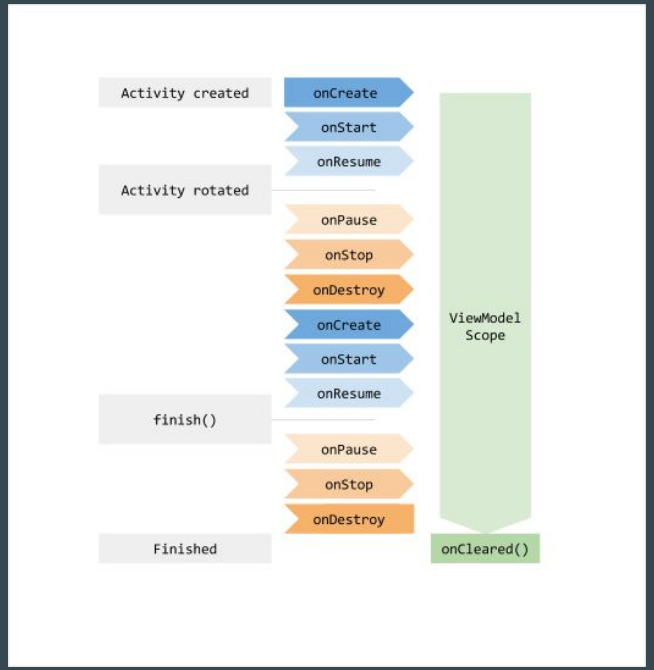
Grundlagen - Context

	Application	Activity	Service	ContentProvider	BroadcastReceiver
Show a Dialog	NO	YES	NO	NO	NO
Start an Activity	NO[1]	YES	NO[1]	NO[1]	NO[1]
Layout Inflation	NO[2]	YES	NO[2]	NO[2]	NO[2]
Start a Service	YES	YES	YES	YES	YES
Bind to a Service	YES	YES	YES	YES	NO
Send a Broadcast	YES	YES	YES	YES	YES
Register BroadcastReceiver	YES	YES	YES	YES	NO[3]
Load Resource Values	YES	YES	YES	YES	YES

Grundlagen - App-Architektur

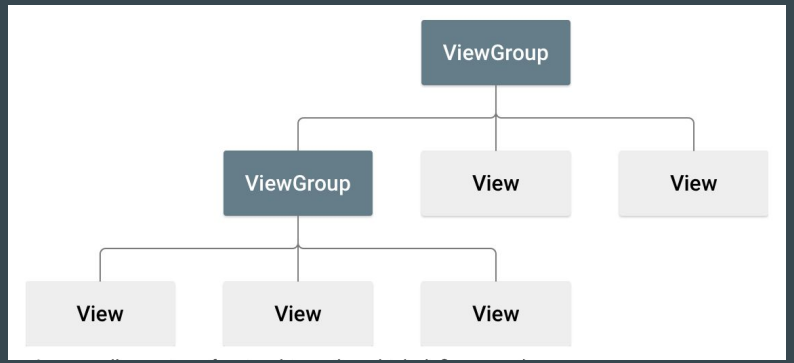
- Activity -> MVP -> MVVM
- Ohne Architektur: alles passiert in der Activity
- Einführung von MVVM mit Arch-Components
- MVVM
  - Unabhängig vom Lifecycle der Activity
  - Daten die den aktuellen Zustand des Screens darstellen
  - Bereitstellung durch Factory

ViewModel



Grundlagen - User-Interface

- UI wird in XML definiert
- jeder Screen referenziert eine Layout-Ressource



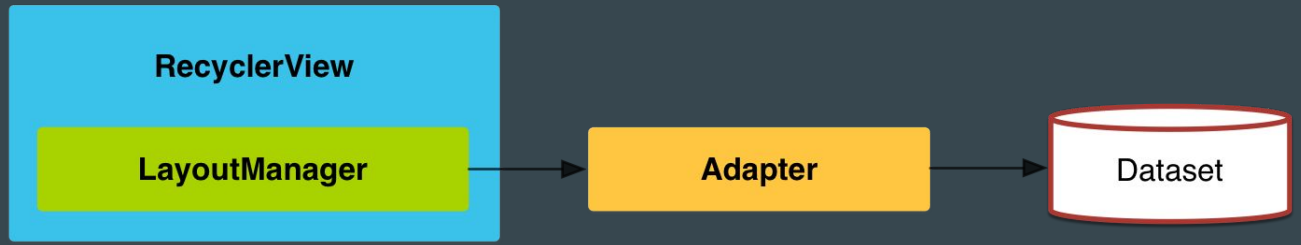
- View: Basisklasse für alle UI-Elemente
  - TextView
  - ImageView
  - ScrollView
  - Button
- ViewGroup: Gruppierung von mehreren Views
  - FrameLayout
  - LinearLayout
  - RelativeLayout
  - ConstraintLayout

Grundlagen - Listen

- verschiedene Listen existieren:
  - ListView
  - GridView
  - RecyclerView
- Listen bekommen Daten über Adapter (z.B. ArrayAdapter für ListView)
- Adapter verknüpfen Listeneinträge mit einem Layout

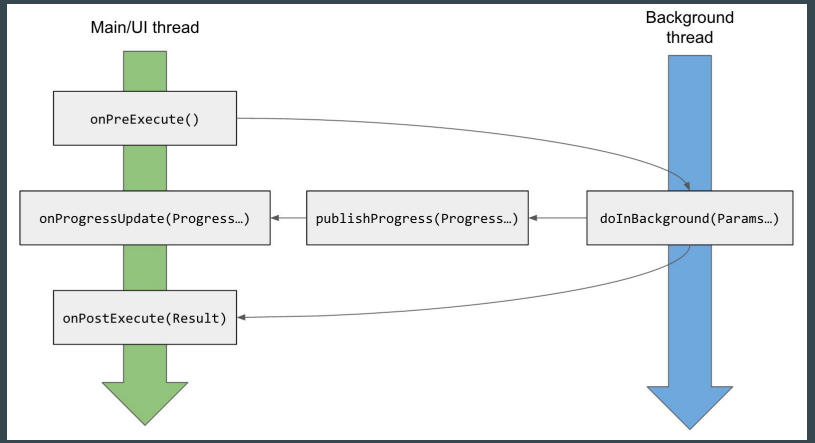
RecyclerView

- Layout der Elemente ist austauschbar
  - LinearLayoutManager
  - GridLayoutManager
- nutzt ViewHolder pattern -> Wiederverwendung von Elementen

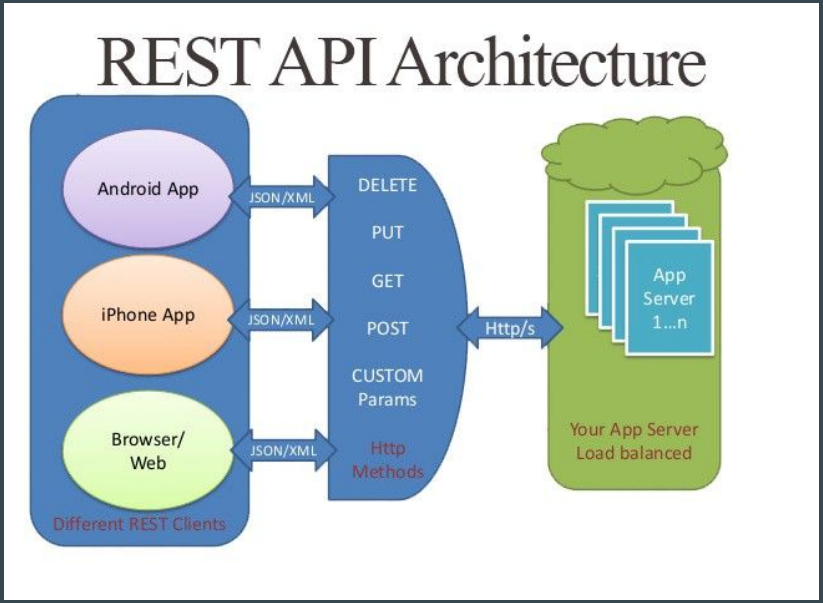


Netzwerk - Asynchronität

- UI wird auf dem Main-Thread gezeichnet (60 fps)
- Längere Berechnungen führen zu ANR (Android not responding)
  -> andere Threads erforderlich
- Netzwerk-Abfragen dürfen nicht auf Main-Thread sein
  -> NetworkOnMainThreadException
- Möglichkeiten:
  - Java Concurrency -> eigene Threads
  - AsyncTask
  - IntentService
  - RxJava
  - Kotlin Coroutines



Netzwerk - RESTful Webservices



Retrofit

- REST-Client für Android (Square)
- Request werden in Interface definiert
- Methoden als Annotation definiert

Netzwerk - Retrofit

```
public interface GitHubService {
  @GET("users/{user}/repos")
  Call<List<Repo>> listRepos(@Path("user") String user);
}
```

```
Retrofit retrofit = new Retrofit.Builder()
    .baseUrl("https://api.github.com/")
    .build();

GitHubService service = retrofit.create(GitHubService.class);
```

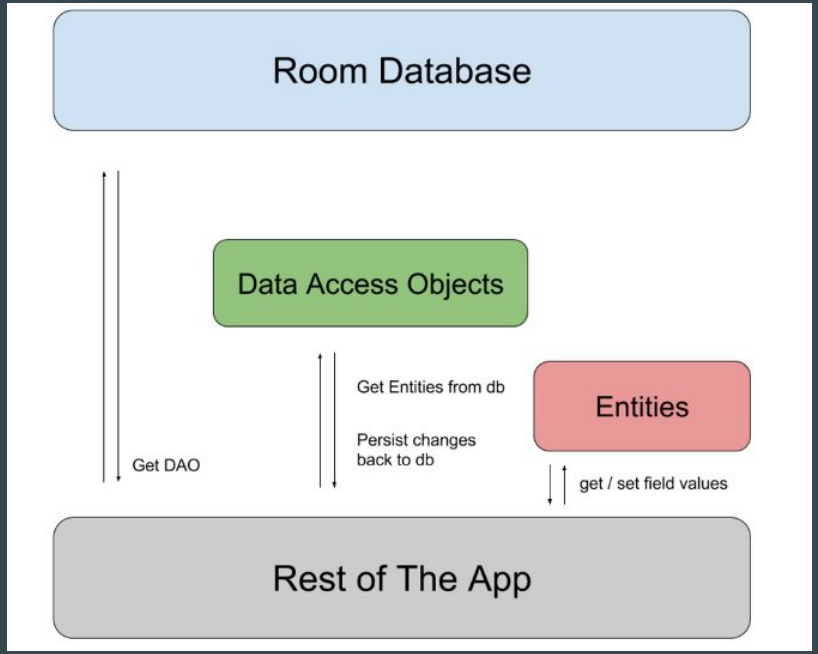
```
Call<List<Repo>> repos = service.listRepos("octocat");
```

Persistenz - Übersicht

- SharedPreference
  - Speichern von primitiven Daten als Key-Value Paar (String, Int, Long, Boolean)
  - Einfache XML-Dateien, verwaltet vom Android-System
- Dateisystem
  - Interner Speicher: nur die App hat Zugriff
  - Externer Speicher: alle Apps haben Zugriff (Permission vorausgesetzt)
- Datenbank
  - SQLite: platzsparend SQL Variante, speziell für Mobilgeräte
    android.database.sqlite, GreenDAO, Room
  - NoSQL: schnelle Lese- und Schreibprozesse
    Realm, ObjectBox

Persistenz - Room

- SQLite ORM Implementierung von Google
- Komponenten:
  - Entities: Objektrepräsentation der Tabellen
  - Dao: definiert alle CRUD operationen
  - Database: Zugriff alle Dao's

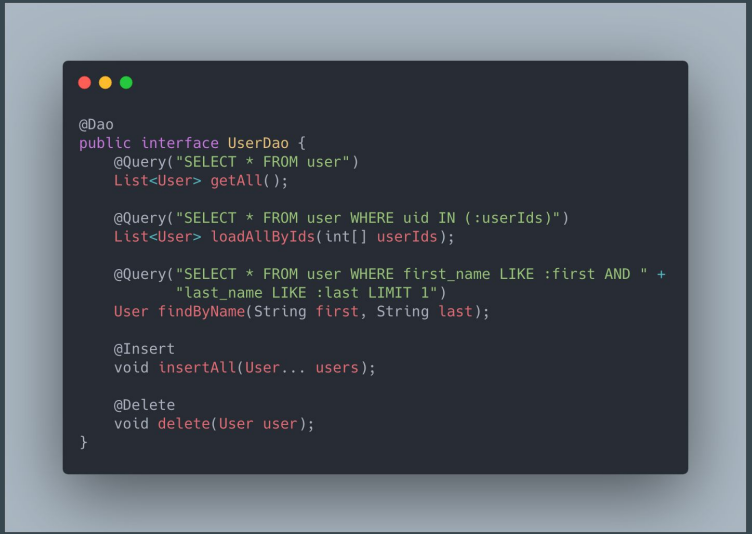


Room - Entity, Dao

```java
@Entity
public class User {
    @PrimaryKey
    public int uid;

    @ColumnInfo(name = "first_name")
    public String firstName;

    @ColumnInfo(name = "last_name")
    public String lastName;
}
```



```java
@Dao
public interface UserDao {
    @Query("SELECT * FROM user")
    List<User> getAll();

    @Query("SELECT * FROM user WHERE uid IN (:userIds)")
    List<User> loadAllByIds(int[] userIds);

    @Query("SELECT * FROM user WHERE first_name LIKE :first AND " +
           "last_name LIKE :last LIMIT 1")
    User findByName(String first, String last);

    @Insert
    void insertAll(User... users);

    @Delete
    void delete(User user);
}
```

Room - Database

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

```
AppDatabase db = Room.databaseBuilder(getApplicationContext(),
        AppDatabase.class, "database-name").build();
```