Portál pre odborné publikovanie ISSN 1338-0087

Platforma Android

Vanderka Marián · Informačné technológie, Študentské práce

20.05.2011

Android predstavuje softwarovú sadu pre mobilné zariadenia, ktorá pozostáva z operačného systému, middlewareu a samotných aplikácií. Nástroje a API (application programming interface) potrebné na vývoj aplikácií pre Android platformu sú obsiahnuté v Android SDK (software development kit). Používa sa programovací jazyk Java, ale je možné vytvárať aj vlastné knižnice v jazyku C/C++.

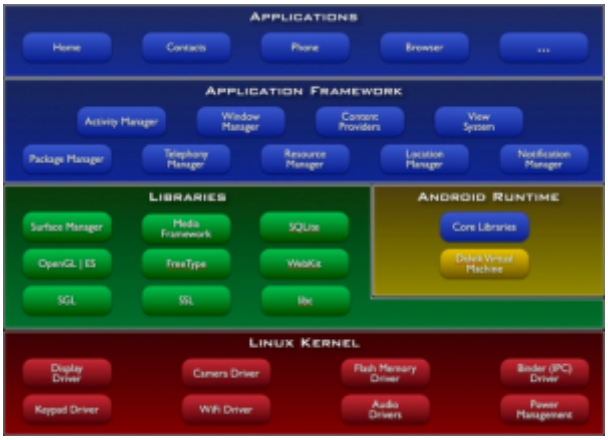
Základné vlastnosti:

- Architektúra aplikácií umožňuje jednoducho vymieňať a znova využiť jednotlivé komponenty
- Využíva sa Dalvik Virtual Machine, ktorý je optimalizovaný pre mobilné zariadenia
- Vstavaný webový browser založený na voľne dostupnom WebKit engine
- Optimalizovaná grafika založená na vlastných knižniciach pre 2D grafiku, zatiaľ čo 3D grafika zakladá na špecifikácii OpenGL ES 1.0 a OpenGL ES 2.0 od verzie Android 2.0
- SQLite databázový systém
- Podpora bežných multimediálnych formátov (MPEG4, H.264, MP3, AAC, AMR, JPG, PNG, GIF)
- GSM telefónia
- Bluetooth, EDGE, 3G a WiFi
- Kamera, GPS, kompas, akcelerometre a iné senzory
- Bohaté vývojové prostredie, ktoré obsahuje emulátor zariadenia, nástroje pre debugging, správu výkonu a pamäte a plugin pre Eclipse IDE

(pre správne fungovanie všetkých častí je potrebný príslušný hardware)

Architektúra Androidu

Na nasledujúcom diagrame môžeme sledovať hlavné komponenty Android systému, ktoré sú následne stručne opísané.



Aplikácie

Android v sebe obsahuje sadu základných aplikácií vrátane emailového klienta, SMS aplikácia, kalendár, mapy, prehliadač, kontakty a iné. Všetky aplikácie sú napísané v Java-e

Aplikačný rámec

Android umožňuje vývojárom vďaka otvorenej platforme budovať kvalitné aplikácie. Môžu využívať hardware na zariadení, lokačné služby a informácie, spúšťať procesy na pozadí, nastavovať alarmy, notifikácie a mnoho ďalšieho. Majú plný prístup k rovnakým API ako používajú systémové aplikácie. Architektúra je nastavená tak aby bolo možné jednoducho znova používať jednotlivé komponenty – aplikácie môžu zverejniť svoje schopnosti aby ich ostatné mohli využiť pre seba pričom sa kladie veľký dôraz na bezpečnosť. Rovnaký mechanizmus umožňuje upravovanie a vytváranie vlastných komponent.

Medzi základné komponenty patria napr.:

- Bohatá a rozšíriteľná sada okien (Views), ktoré umožňujú vytvoriť GUI aplikácie pomocou rôznych zoznamov, textových polí, tlačítok, mriežok, tabov a taktiež aj vstavateľného webového prehliadača
- Na prístup ku dátam z iných aplikácií (napr. Kontakty, SMS…), prípadne zverejnenie vlastných dát sa používa komponenta Content Provider
- Resource Manager umožňuje jednoduchý prístup ku zdrojom nezahrnutým v kóde ako sú lokalizované texty, grafika, zvuky a rôzne .xml súbory
- Pomocou Notification Manager-u získame prístup ku rozťahovateľnému panelu s notifikáciami a statusmi
- Dôležitou komponentou je Activity Manager ktorý zabezpečuje životný cyklus aplikácií a prepínanie medzi nimi

Knižnice

Android obsahuje sadu C/C++ knižníc ktoré sú využívané komponentami systému. Tieto sú samozrejme taktiež dostupné vývojárom k dispozícii. Medzi hlavné patria:

- Systémová C knižnica – BSD-odvodená implementácia štandardnej C systémovej knižnice (libc), upravená pre vstavané linuxové zariadenia

- Multimediálne knižnice založené na OpenCORE od PacketVideo; tieto umožňujú prehrávanie a nahrávanie audia, videa ako aj statických obrázkov vo formátoch ako sú MPEG4, H.264, MP3, AAC, AMR, JPG a PNG
- Surface Manager – poskytuje prístup k obrazovému podsystému a skladá 2D a 3D grafiku z viacerých aplikácií
- LibWebCore – slúži ako engine pre zabudovaný webový prehliadač a taktiež pre zabudovateľné okno (View) prehliadača
- SGL – engine pre 2D grafiku
- Knižnice pre 3D grafiku sú založené na OpenGL ES 1.0 a 2.0 API, pričom využívajú buď hardwarovú 3D akceleráciu ak je dostupná prípadne vysoko optimalizovanú softwarovú akceleráciu
- Na vykresľovanie bitmapových a vektorových fontov slúži knižnica FreeType
- SQLite poskytuje kvalitnú a odľahčenú relačný databázový systém dostupný pre všetky aplikácie

Android Runtime

Android obsahuje sadu základných knižníc ktoré poskytujú väčšinu funkčnosti dostupnej v základných knižniciach jazyka Java. Každá aplikácia beží vo vlastnom procese s vlastnou inštanciou Dalvik virtuálneho stroja. Dalvik bol napísaný tak aby zariadenie zvládlo efektívne spúšťať viacero VS. Spúšťacie súbory pre Dalvik VS sú vo formáte .dex (Dalvik Executable) optimalizovanom pre čo najmenšiu spotrebu pamäte. VS je založený na registroch a spúšťa triedy skompilované pomocou kompilátora jazyka Java, ktoré boli prevedené do formátu .dex pomocou dostupného “dx” nástroja. Dalvik VS je postavený na linuxovom kerneli zabezpečujúcom nízko úrovňovú funkčnosť ako sú thready a správa pamäte.

Linux Kernel

Android využíva verziu 2.6 pre základné systémové služby ako je bezpečnosť, správa pamäti, správa procesov, sieťové prostriedky a model ovládačov. Kernel slúži aj ako abstraktná vrstva medzi hardwarom a softwarovou vrstvou Aplikácie pre Android sú napísané v jazyku Java. Nástroje obsiahnuté v Android SDK skompilujú kód zároveň s dátami a zdrojmi do jedného balíka, archívu s koncovkou .apk. Tento súbor sa považuje za celú aplikáciu a slúži na inštaláciu v zariadení. Po inštalácii na zariadenie aplikácia “žije” vo vlastnom bezpečnom sandboxe.

- Android OS je linuxový systém kde je každá aplikácia považovaná za samostatného užívateľa
- Každá aplikácia má priradené jedinečné linuxové užívateľské ID (pou69va ho len systém a je neznáme pre aplikáciu). Systém potom nastaví potrebné práva pre všetky súbory v rámci aplikácie aby jedine táto mala k nim prístup pomocou svojho ID
- Procesy aplikácie bežia v samostatnom VS, takže kód aplikácie je oddelený od ostatných aplikácií
- Každá aplikácia beží vo vlastnom procese. Tento proces sa spúšťa keď je potrebné spustiť jednotlivé komponenty a zatvorí sa, keď už nie je potrebný alebo systém potrebuje obnoviť pamäťové prostriedky

Týmto spôsobom Android implementuje princíp najnižších práv, to znamená že každá

aplikácia má len prístup ku tým komponentom ktoré nevyhnutne potrebuje na svoju prevádzku a nič viac. Vytvára sa dobre zabezpečené prostredie v ktorom aplikácie nemajú prístup ku častiam systému kde nemajú potrebné práva. Avšak je tu niekoľko spôsobov ako aplikácie dokážu zdieľať s ostanými dáta prípadne pristupovať ku systémovým prostriedkom:

- Je možné aby aplikácie zdieľali rovnaké linuxové užívateľské ID, čo im umožní pristupovať k súborom tej druhej. Za účelom ušetrenia systémových zdrojov je možné aby tiež bežali v rovnakom procese a zdieľali spoločný VS (musia byť podpísané rovnakým certifikátom)
- Aplikácia môže požiadať o práva pre prístup ku dátam ako sú kontakty, SMSky, obsah na SD karte, kamera, bluetooth a mnohé iné. Tieto je potrebné potvrdiť pri inštalácii.

Aplikácia v Android pozostáva s niekoľkých aplikačných komponent (activities, services, content providers, a broadcast receivers). Každá komponenta vykonáva inú úlohu v celkovom správaní sa aplikácie a zároveň je možné ich spúšťať samostatne, taktiež aj z iných aplikácií. Všetky komponenty a požiadavky na SW a HW musia byť deklarované v súbore ktorý sa nazýva manifest. Zdroje mimo kódu (obrázky, reťazce, .xml súbory….) by mali obsahovať aj alternatívy pre rôzne konfigurácie cieľových zariadení (reťazce vo viacerých jazykoch, iné rozloženie na základe rozlíšenia….).

Activities

Acitvity reprezentuje jednu obrazovku s UI (user interface – užívateľské rozhranie). Napríklad emailová aplikácia môže obsahovať jednu ktorá zobrazuje došlé emaily, ďalšiu ktorá slúži na písanie správ a ďalšiu na ich čítanie. Aj keď spolu tvoria jeden celok v hľadiska používateľského rozhrania, navzájom sú nezávislé. Toto umožňuje ostatným aplikáciám ich spúšťanie, ak to spúšťaná aplikácia dovoľuje. Napríklad aplikácia pre kameru môže pristupovať ku časti ktorá iba posiela email za účelom poslania fotky. Implementovaná je ako podtrieda triedy Activity.

Serivces

Sú to komponenty ktoré bežia na pozadí a slúžia na spracovanie dlho trvajúcich operácii alebo vykonávajú úlohy pre Ine procesy. Nemajú žiadne UI. Typickým príkladom je prehrávač hudby ktorý beží na pozadí zatiaľ čo užívateľ je v inej aplikácii alebo sťahovanie dát bez toho aby bolo UI blokovane. Ine komponenty , najčastejšie activity sa ich môžu spúšťať, pripadne sa k nim pripojiť pre lepšiu interakciu. Implementuje sa trieda Service.

Content providers

Slúžia na správu zdieľaných dát aplikácie. Dáta môžu byt uložené v súborovom systéme, SQLite databáze, na webe alebo hocijakom inom úložnom priestore kde ma aplikácia prístup. Pomocou tejto komponenty môžu potom aplikácie dopytovať a meniť(ak je to povolene) dostupne dáta. Systém napríklad pomocou tejto komponenty spravuje údaje o kontaktoch užívateľa. Každá aplikácia s potrebnými právami môže potom pristupovať k určitej časti dát a citát a zapisovať informácie o konkrétnom kontakte. Je vhodne ich použiť aj v prípade že dáta sú privátne a nie sú zdieľané. Hlavná trieda sa vola ContentProvider a musí byt implementovaná aj štandardná sada API ktorá umožní ostatným aplikáciám narábať s dátami.

Broadcast receivers

Tieto komponenty reagujú na oznámenia vysielane pre cely systém. Mnoho sprav pochádza zo systému, napríklad oznámenie a slabej baterke, vypnutí displeja atd. Aplikácie tieto správy môžu tiež vysielať, napríklad oznam o stiahnutí dát a ich prístupnosť pre ostatne aplikácie. Aj keď tiež nemajú žiadne UI, môžu vytvárať notifikácie na statusovom paneli čím upozorňujú že sa nastala udalosť ktorá vyžaduje pozornosť. Najčastejšie však slúžia len ako akési braný pre ostatne komponenty a vykonávajú minimálne množstvo operácii. Na základe nastanej udalosti napríklad spustia servise ktorý vykoná potrebne operácie. Implantovaná trieda je BroadcastReceiver a oznamy sú doručované ako Intent objekty.

Aktivovanie komponent

Activities, services, a broadcast receivers sa spúšťajú pomocou asynchrónnej správy ktorá sa nazýva intent. Intenty zvezujú jednotlivé komponenety medzi sebou počas behu systému pričom nezáleží v akej aplikácii sa nachádzajú . Vytvárajú sa pomocou objektu Intent, definujúci správu ktorou sa aktivuje žiadaná konkrétna komponenta alebo len špecifický typ komponenty – explicitný alebo implicitný intent. V správe môže byť obsiahnutá akcia aká sa má vykonať ako aj príslušné dáta k tejto akcii.

Pre prístup ku content provider-u sa používajú požiadavky od ContentResolver-u. Tieto medzi sebou priamo komunikujú. K dátam sa pristupuje pomocou metód volaných na objekt ContentResolver-u, čím sa vytvára abstraktná vrstva medzi zdrojom dát (content provider) a komponentami žiadajúcimi dáta.

Pomocou implicitných intentov môžeme spúšťať komponenty bez toho aby sme vedeli ich názov. Takýto intent musí obsahovať akciu ktorá sa má vykonať. Systém následne pomocou intent-filter-ov definovaných v aplikáciach vyhodnotí zoznam vhodných komponent ktoré sú danú akciu schopné splniť. Komponenty sú deklarované v manifestovom súbore spolu s údajmi o intentoch ktoré chcú prijímať.

Manifestový súbor

Systém pred tým ako môže spustiť komponentu aplikácie musí vedieť o jej existencii prečítaním AndroidManifest.xml súboru nachádzajúceho sa pri aplikácii. Tu musia byť deklarované všetky komponenty a je umiestnený v koreňovej zložke projektu. Okrem toho slúži ešte napríklad na definovanie práv o ktoré aplikácia žiada (prístup na internet, čítanie kontaktov…), deklarovanie minimálnej verzie API, deklarovanie HW a SW funkcii ktoré vyžaduje (kamera, BT, WiFi, touchsreen…), zoznam externých knižníc ktoré využíva (knižnica Google Maps) a iné.

Deklarovanie požiadaviek je z hľadiska konečného užívateľa dôležitý krok. Android sa používa na veľkom počte rôznych zariadení, ktoré sa od seba líšia hardwarom ale aj verziou Androidu. Tieto informácie pre samotnú aplikáciu nie sú potrebné a sú len informačného charakteru avšak služby ako Goole Market tieto informácie používa na filtrovanie vhodných aplikácii keď hľadáme aplikácie priamo zo zariadenia. Je dobré aby aplikácia bola dostupná len pre tie zariadenia ktoré spĺňajú deklarované požiadavky. Medzi hlavné vlastnosti o ktorých by sme mali pri návrhu aplikácie uvažovať sú:

- veľkosť displeja a jeho hustota; je vhodné rozloženie UI vytvoriť pre viacero možných veľkostí (small, normal, large, extra large) ako aj grafiku upraviť pre rôzne hustoty (low, medium, high, extra high) a nakoniec definovať ktoré hodnoty aplikácia podporuje
- typ vstupného zariadenia; trackball, touchsreen, klávesnica….
- možnosti zariadenia; napr. svetelný senzor, BT, WiFi, verzia OpenGL, nikdy by sme nemali predpokladať že každé zariadenie podporuje nami vyžadovanú možnosť
- verzia API; každá nová verzia Androidu využíva dodatočné API, čiže na starších platformách aplikácia nemusí fungovať (napr. Android 2.3 je API verzia 9)

Zdroje mimo kód

Na definovanie animácií, menu, štýlov, rozloženia, reťazcov, farieb… sa používajú .xml súbory. Vďaka tomu je možné optimalizovať aplikáciu pre viacero konfigurácii, jazykových verzií, aktuálnu orientáciu zariadenia… bez toho aby sme museli meniť samotný kód aplikácie. Všetkým zdrojom ktoré definujeme, SDK pomocou svojich build nástrojov priradí jedinečné číselné ID, pomocou ktorého môžeme potom k nim pristupovať v kóde. Pomocou kvalifikátorov (špecifický krátky text) za menom adresára so zdrojmi Android vie kedy čo použiť, napr. res/values-sk/ reprezentuj hodnoty relevantné pre užívateľa zo slovenska.