UNIVERZA V LJUBLJANI
FAKULTETA ZA RAČUNALNIŠTVO IN INFORMATIKO

Andraž Podobnik

Platforma za spremljanje poteka košarkarskih tekem

DIPLOMSKO DELO
NA UNIVERZITETNEM ŠTUDIJU

Mentor: doc. dr. Rok Rupnik

Ljubljana, 2013

Univerza *v Ljubljani*
Fakulteta *za računalništvo in informatiko*

Tržaška 25
1001 Ljubljana, Slovenija
telefon: 01 476 84 11
01 476 83 87
faks: 01 426 46 47
01 476 87 11
www.fri.uni-lj.si
e-mail: dekanat@fri.uni-lj.si

Št. naloge: 01902/2013
Datum: 06.02.2013

Univerza v Ljubljani, Fakulteta za računalništvo in informatiko izdaja naslednjo nalogo:

Kandidat: **ANDRAŽ PODOBNIK**

Naslov: **PLATFORMA ZA SPREMLJANJE POTEKA KOŠARKARSKIH TEKEM**
PLATFORM TO MONITOR BASKETBALL MATCHES

Vrsta naloge: Diplomsko delo univerzitetnega študija

Tematika naloge:

Zasnujte in izdelajte načrt za platformo, ki bo omogočala vnos podatkov o dogodkih na košarkaških tekmah in obvladovanje statistike tekem. V sklopu platforme naj bosta poleg strežnika tudi mobilna aplikacija za učinkovit vnos dogodkov ter mobilna aplikacija za spremljanje statističnih podatkov o tekmi. Na podlagi načrta razvijte strežnik in mobilni aplikaciji.

Mentor:

doc. dr. Rok Rupnik

Fakulteta za računalništvo in informatiko • UNIVERZA V LJUBLJANI • 2

Dekan:

prof. dr. Nikolaj Zimic

IZJAVA O AVTORSTVU

diplomskega dela

Spodaj podpisani **Andraž Podobnik**,

z vpisno številko **63070187**,

sem avtor diplomskega dela z naslovom:

Platforma za spremljanje poteka košarkarskih tekem

S svojim podpisom zagotavljam, da:

- sem diplomsko delo izdelal/-a samostojno pod mentorstvom
  doc. dr. Roka Rupnika
- so elektronska oblika diplomskega dela, naslov (slov., angl.), povzetek (slov., angl.) ter ključne besede (slov., angl.) identični s tiskano obliko diplomskega dela
- soglašam z javno objavo elektronske oblike diplomskega dela v zbirki "Dela FRI".

V Ljubljani, dne 7.3.2013 Podpis avtorja/-ice:

Zahvala

Na tem mestu bi se zahvalil vsem, ki so me podpirali na poti do diplome. Posebna zahvala gre moji družini in punci Mojci za podporo in spodbudo ob premagovanju vseh mogočih ovir, zahvalil bi se tudi mentorju doc. dr. Roku Rupniku za pomoč in nasvete.

Kazalo

Povzetek **1**

Abstract **2**

1 Uvod **3**

2 Uporabljena orodja in tehnologije pri razvoju sistema **5**

2.1 Android 5

2.1.1 Različice operacijskega sistema Android 6

2.1.2 Arhitektura 6

2.1.3 Aplikacija in njen življenjski cikel 11

2.1.4 Shranjevanje podatkov 15

2.2 Google Cloud Messaging - GCM 17

2.2.1 GCM storitev 19

2.2.2 Arhitekturni pregled GCM storitve 21

2.2.3 Vloga aplikacijskega strežnika 25

2.3 PHP 31

2.4 MySQL 31

2.5 Razvojna orodja 32

2.5.1 Eclipse 32

2.5.2 Android SDK 35

3 Mobilna aplikacija **36**
3.1 Načrtovanje in razvoj mobilne aplikacije 36
3.1.1 Načrtovanje podatkovne baze 38
3.1.2 Razvoj aktivnosti za pisanje statistike ter pošiljanje sporočil strežniku 41
3.1.3 Razvoj aktivnosti za prejemanje sporočil od GCM strežnika 44

4 Strežniški del **48**

5 Zaključek **51**

Seznam slik **53**

Seznam tabel **54**

Literatura **55**

Seznam uporabljenih kratic in simbolov

2D - Two Dimensional; dvo-dimenzionalno

3D - Three Dimensional; tri-dimenzionalno

ADT - Android Development Tools; razvojna orodja za operacijski sistem Android

API - Application Programming Interface; programski vmesnik

GCM - Google Cloud Messaging; Googlova storitev za pošiljanje sporočil

GPS - Global Positioning System; sistem globalnega določanja lege

HTML - Hypertext Markup Language; označevalni jezik za oblikovanje večpredstavnostnih vsebin

HTTP - Hypertext Transfer Protocol; protokol za prenos informacij na spletu

HTTPS - Hypertext Transfer Protocol Secure; protokol za varen prenos informacij na spletu

ID - Identifier; enolični identifikator

IMEI - International Mobile Station Equipment Identity - Unikatna številka za identifikaticijo mobilnih naprav

JSON - JavaScript Object Notation; preprost format za izmenjavo podatkov

RSS - Rich Site Summary; zgoščeni povzetek strani

SDK - Software Development Kit; paket programskih orodij za razvoj aplikacij

SMS - Short Message Service; sistem za pošiljanje krajših besedilnih sporočil z mobilnim telefonom

SQL - Structured Query Language; strukturiran povpraševalni jezik za delo s podatkovnimi zbirkami

WiFi - tehnologija brezžičnega prenosa podatkov preko računalniškega omrežja

XML - Extensible Hypertext Markup language; format podatkov za izmenjavo strukturiranih dokumentov v spletu

XMPP - Extensible Messaging and Presence Protocol; komunikacijski protokol na osnovi XML

Povzetek

V diplomski nalogi bomo predstavili sistem, ki uporabniku omogoča pisanje statistike košarkarskih tekem in omogoča spremljanje rezultatov v živo tistim, ki jih rezultat določene tekme zanima, vendar ne morejo biti prisotni na tekmi. Spremljanje rezultata omogoča Android aplikacija.

V diplomskem delu se najprej spoznamo z operacijskim sistemom Android ter storitvijo GCM, v nadaljevanju naloge opišemo razvoj pomembnejših delov sistema, ki smo ga naredili. Prvi del sistema je aplikacija za mobilne naprave z Androidom, ki omogoča dve stvari. Prva je pisanje statistike, druga pa spremljanje rezultatov tekem v živo. Statistika se shranjuje na mobilno napravo, ki si jo je mogoče pogledati za vse tekme nazaj. Za spremljanje rezultatov smo uporabili storitev GCM, ki ponuja PUSH način komunikacije. Rezultati tekem se na mobilnih napravah prikazujejo v statusni vrstici, kjer se izpiše rezultat in imeni obeh moštev, ki igrata. Drug del sistema je strežnik, ki na eni strani dobiva sporočila (tj. rezultate tekem) od tistega, ki na tekmi s pomočjo aplikacije piše statistiko in prejeta sporočila pošilja naprej tako, da jih dobijo vsi, ki si želijo rezultate tekem spremljati. V zaključku smo podali nekaj idej za izboljšavo in nadaljnji razvoj sistema.

Ključne besede:

Android, storitev GCM, mobilna aplikacija, košarka

Abstract

In the thesis we will present a system which enables writing statistics of basketball matches and monitor live results of matches for those users, who are not present on the matches. Monitoring of live result will be enabled through the Android application.

In the first part of thesis we describe operating system Android and GCM service, continuing with the main part where we describe the development of important parts of the system we have made. The first part of the system is an application designed for mobile devices with Android, which allows two things. The first is writing statistics and the second is monitoring live results of other matches. Statistics are stored on the mobile device for the purpose to view any match at any time user wants. For the purpose of monitoring the results, GCM service, which provides PUSH notification, is used. The results of matches appear in the status bar of mobile devices and consist of result with both teams names. The second part of the system is the server that receives messages (i.e. the results of matches) from users who write statistics using application on one hand and on the other hand sends messages on. Regardingly everybody who is interested in results can get them. In conclusion some ideas for improvement and further development of the system are given.

Key words:

Android, GCM service, mobile application, basketball

Poglavje 1

Uvod

Razvoj mobilnih naprav je danes vse hitrejši, z vsako novo generacijo pa se njihova uporabnost še povečuje. Zagotovo k temu veliko pripomorejo tudi operacijski sistemi, ki tečejo na njih. Glavna igralca na tem področju sta trenutno Apple in Google, ki se s svojima iOS in Android sistemoma borita za prevlado na trgu mobilnih operacijskih sistemov. Zavedata se, da si uporabniki želijo imeti vse potrebne informacije na dosegu roke kadarkoli in kjerkoli. Operacijska sistema dopolnjujejo oziroma izpopolnjujejo tudi aplikacije, brez katerih zagotovo ne bi zaživela tako kot sta. Za oba sistema je na voljo že preko sedemsto tisoč aplikacij.

Zamisel za izdelavo sistema oziroma aplikacije sem dobil lansko leto, ko sem začel igrati košarko pri košarkarskemu klubu, ki igra v nižji slovenski ligi. V višjih ligah podrobnejša statistika tekme in spremljanje rezultata tekme v živo, tudi če nismo na tekmi, nista problem. Uredne osebe ju namreč morajo obvezno pisati. V omenjeni ligi pa naletimo na problem. Nihče ne more spremljati rezultata tekme v živo, če ga ni v dvorani, kjer se tekma odvija. To je bil glavni razlog, da sem se odločil, da bom napisal aplikacijo, ki bo ta problem rešila. Jasno je namreč, da število pametnih telefonov hitro raste, poleg tega ima mobilni telefon čina ljudi vedno s sabo. Tako bo lahko nekdo, ki bo tekmo gledal v živo, pisal statistiko s pomočjo aplikacije, s tem pa bo omogočil drugim

spremljanje rezultata tekme ne glede na to, kje bodo v času tekme. Glede na to, da sam uporabljam mobilni telefon z operacijskim sistemom Android, bom razvil aplikacijo za Android mobilne naprave.

Poglavje 2

Uporabljena orodja in tehnologije pri razvoju sistema

2.1 Android

Android je na Linuxu temelječ operacijski sistem in je v osnovi namenjen mobilnim napravam z zaslonom na dotik, tj. pametnim telefonom (ang. smartphones) in tabličnim računalnikom (ang. tablet computers). Gre za odprtokodni projekt, ki ga razvija Odprto združenje mobilnih naprav (ang. Open Handset Alliance) z Googlom na čelu. Google kodo operacijskega sistema Android objavlja pod Apache licenco, kar pomeni, da se lahko izvorno kodo Androida poljubno spreminja.

Android ima veliko skupnost razvijalcev, ki zanj razvijajo aplikacije. Aplikacije dodajajo napravam nove funkcionalnosti in s tem boljšo uporabniško izkušnjo. Osnovni način razvijanja aplikacij je v programskem jeziku Java, sam izgled aplikacije pa definiramo z XML datotekami.

2.1.1 Različice operacijskega sistema Android

Do leta 2013 je izšlo že precejšnje število različic operacijskega sistema Android. Z vsako različico je sistem postajal boljši in uporabnejši. Spodnja tabela prikazuje različice sistema Android od 1.6 naprej, saj različice starejše od te, nimajo več omembe vredne razširjenosti med uporabniki. V tabeli 2.1 je prikazana razširjenost posameznih različic operacijskega sistema Android na podatkih, ki jih je Google zbral v 14 dnevnem časovnem obdobju (obdobje se je končalo 3. januarja 2013). Slika 2.1 prikazuje razširjenost posameznih različic še na grafični način.

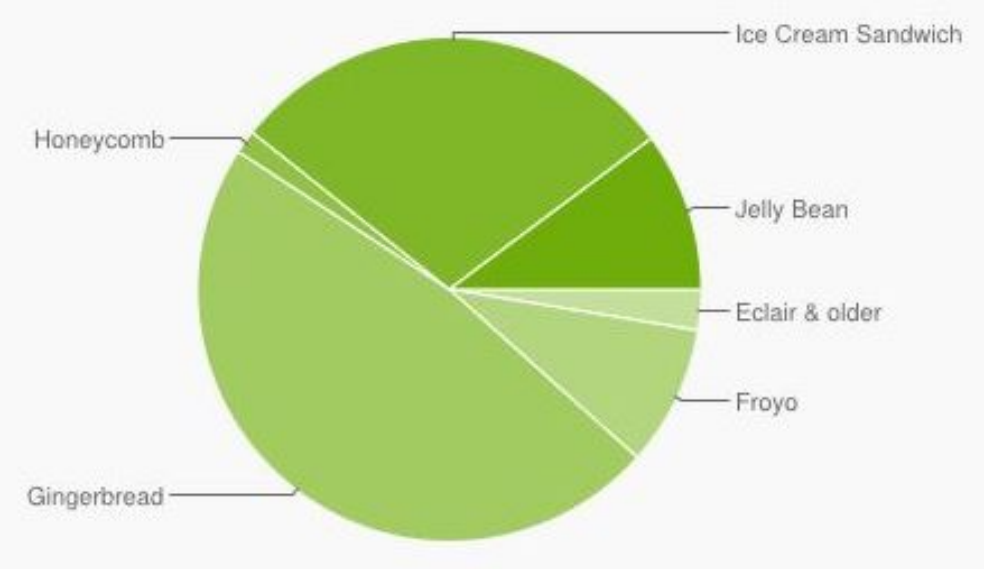
Različica	Kodno ime	API nivo	Razširjenost
1.6	Donut	4	0.2%
2.1	Eclair	7	2.4%
2.2	Froyo	8	9.0%
2.3 - 2.3.2	Gingerbread	9	0.2%
2.3.3 - 2.3.7	Gingerbread	10	47.4%
3.1	HoneyComb	12	0.4%
3.2	HoneyComb	13	1.1%
4.0.3 - 4.0.4	Ice Cream Sandwich	15	29.1%
4.1	Jelly Bean	16	9.0%
4.2	Jelly Bean	17	1.2%

Tabela 2.1: Razširjenost posameznih različic operacijskega sistema Android.

2.1.2 Arhitektura

Sistem Android je sestavljen iz petih osnovnih komponent (glej sliko 2.2):

- aplikacij

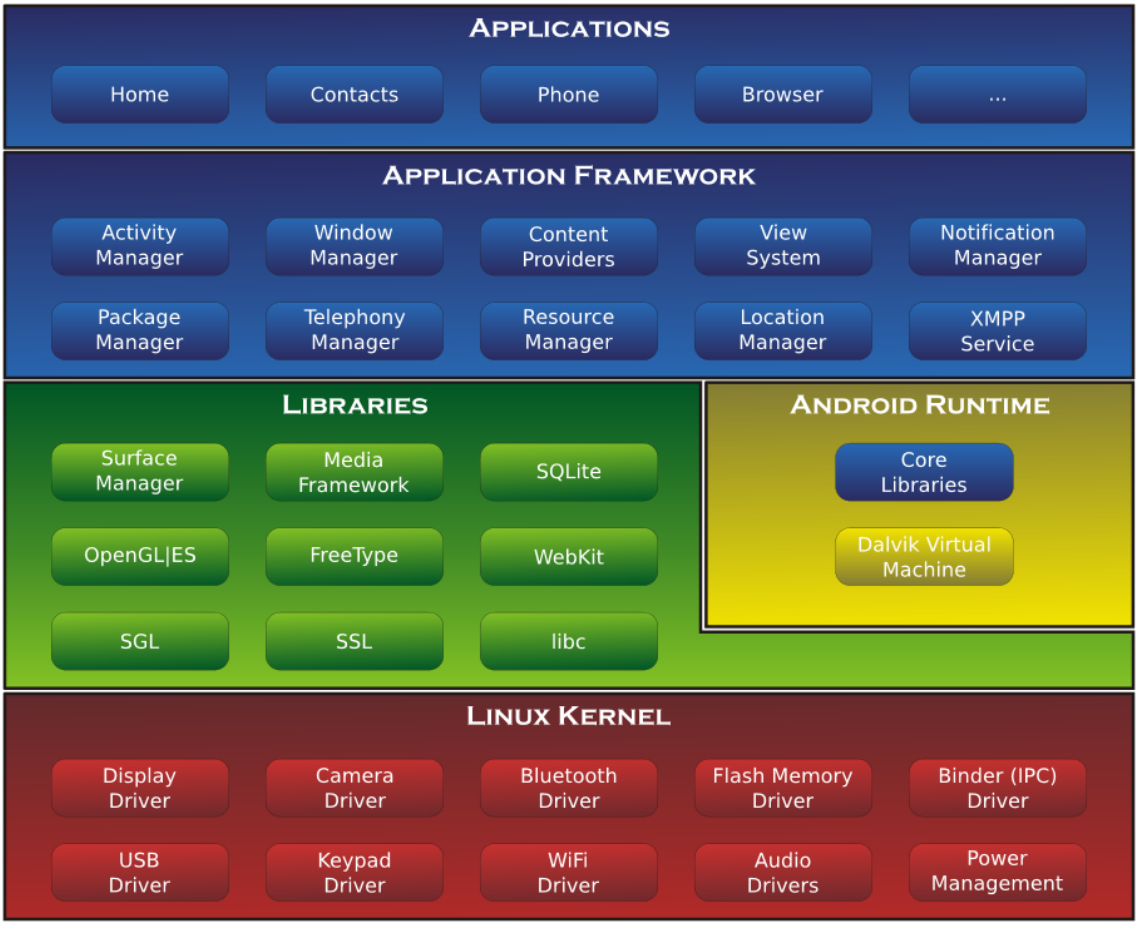


Slika 2.1: Razširjenost posameznih različic med Android napravami.

- aplikacijskega ogrodja
- knjižnic
- zagonskega okolja
- Linux jedra

Aplikacije

S sistemom Android uporabnik dobi osnovni nabor aplikacij, ki ga posamezni izdelovalci mobilnih naprav velikokrat dodatno razširijo. Med njimi so aplikacije kot so: odjemalec elektronske pošte, SMS aplikacija, koledar, zemljevidi, internetni brskalnik, imenik in druge. Tem aplikacijam lahko uporabnik poljubno doda še druge aplikacije, ki jih potrebuje. Najde jih lahko na androidni tržnici Google Play, na spletu ali jih kako drugače naloži na svojo napravo.



Slika 2.2: Arhitektura operacijskega sistema Android.

Aplikacijsko ogrodje

Aplikacijsko ogrodje je v celoti napisano v programskem jeziku Java in vsebuje nabor osnovnih komponent, ki jih med svojim delovanjem uporabljajo vse aplikacije. To velja tako za sistemu priložene aplikacije kot tiste, ki so bile na sistem naložene pozneje. Podrobnejša sestava aplikacijskega ogrodja je sledeča:

- upravitelj aktivnosti (ang. Activity Manager) - upravlja življenjski cikel posameznih aplikacij in omogoča komunikacijo med njimi
- upravitelj paketov (ang. Package Manager) - skrbi za sledenje aplikaci-

jam, nameščenim na telefon

- upravitelj oken (ang. Window Manager) - upravlja okna aplikacije in povzema storitve upravitelja površin (ang. Surface Manager) - glej naslednji sklop Knjižnice
- upravitelj telefonije (ang. Telephony Manager) - vsebuje programske vmesnike, namenjene telefoniji
- ponudnik vsebin (ang. Content Provider) - aplikacijam omogoča dostop do deljenih podatkov, kot so npr. stiki, in omogoča deljenje podatkov med aplikacijami
- upravitelj virov (ang. Resource Manager) - zagotavlja dostop do virov, ki niso del aplikacije, kot so npr. grafični elementi in razporejevalci vsebin
- sistem pogleda (ang. View System) - vsebuje elemente uporabniškega pogleda, kot so npr. gumbi, vnosna polja in seznami
- upravitelj lokacije (ang. Location Manager) - omogoča določanje lokacije s pomočjo GPS-a, WiFi dostopne točke ali omrežja mobilnega operaterja
- upravitelj obvestil (ang. Notification Manager) - skrbi za prikazovanje obvestil v vrstici stanja sistema (ta je na Android mobilnih telefonih povsem na vrhu zaslona)
- servis XMPP - omogoča pošiljanje podatkovnih sporočil drugim uporabnikom Android platforme

Knjižnice

Za uporabo različnih delov sistema ni dovolj le aplikacijsko ogrodje, zato Android vsebuje še zbirko naslednjih programskih knjižnic:

- sistemska knjižnica C - predstavlja za mobilne naprave prilagojeno standardno sistemsko knjižnico libc
- večpredstavnostne knjižnice - temeljijo na knjižnicah paketa PacketVideo OpenCORE in so namenjene predvajanju in snemanju številnih priljubljenih večpredstavnostnih vsebin
- upravitelj površin (ang. Surface Manager) - upravlja dostop do zaslonskega podsistema in sestavlja 2D in 3D sloje aplikacij
- LibWebCore in WebKit - predstavljata vgrajeni internetni brskalnik in podporne komponente
- SGL - predstavlja povezavo z osnovnim slojem 2D grafičnega pogona, 3D knjižnice pa povezavo s podsistemom OpenGL ES, ki skrbi za izris 3D grafike
- knjižnice FreeType - namenjena je prikazu bitne in vektorske pisave
- SQLitea - predstavlja relacijsko zbirko podatkov, ki je vključena v sistem

Zagonsko okolje

Glavni del v zagonskem okolju Android je navidezni stroj Dalvik. Zasnovan je posebej za uporabo v sistemu Android, njegov namen pa je izpolnjevanje zahtev za delovanje v vgrajenem načinu, kjer imamo omejene zmogljivosti baterije, procesorske moči in velikost pomnilnika. Datoteke, ki jih poganja navidezni stroj Dalvik, so s tem optimizirane za minimalno zaseganje sistemskih virov in so v resnici dodatno optimizirane javanske zagonske datoteke. Za čim boljšo izrabo paralelizma in čim manjšo porabo sistemskih virov, navidezni stroj koristi nizkonivojske funkcionalnosti linuxnega jedra.

Linux jedro

Osrednji del Androida predstavlja Linux jedro. Ta predstavlja neke vrste abstrakcijo med strojno opremo in programskim skladom, njegovi odliki pa sta zanesljivost in robustnost. Linux jedro zagotavlja varnost, upravljanje s pomnilnikom, upravljanje s procesi, omrežni sklad in model gonilnikov.

2.1.3 Aplikacija in njen življenjski cikel

Android aplikacija je lahko sestavljena iz kombinacije štirih možnih komponent:

- aktivnost (ang. Activity)
- prejemnik sporočil (ang. Broadcast Receiver)
- storitev (ang. Service)
- upravitelj vsebine (ang. Content Provider)

Podrobneje si bomo pogledali aktivnost in prejemnika sporočil, ki sta pomembni komponenti našega sistema.

Aktivnost

Androidne aplikacije se praviloma zaženejo kot aktivnosti. Aktivnost je ena izmed možnih komponent aplikacije in edina možnost, da aplikacijo zaženemo z uporabniškim vmesnikom. Aktivnost predstavlja posamezni zaslon aplikacije z uporabniškim vmesnikom. Kot primer imamo lahko eno aktivnost, ki vsebuje seznam odigranih tekem in drugo aktivnost, ki vsebuje statistiko izbrane tekme. Kljub temu, da aktivnosti praviloma medsebojno usklajeno podajajo

zvezno uporabniško izkušnjo, so med seboj ločene in lahko do posamezne aktivnosti pridemo na različne načine oz. po različnih poteh.

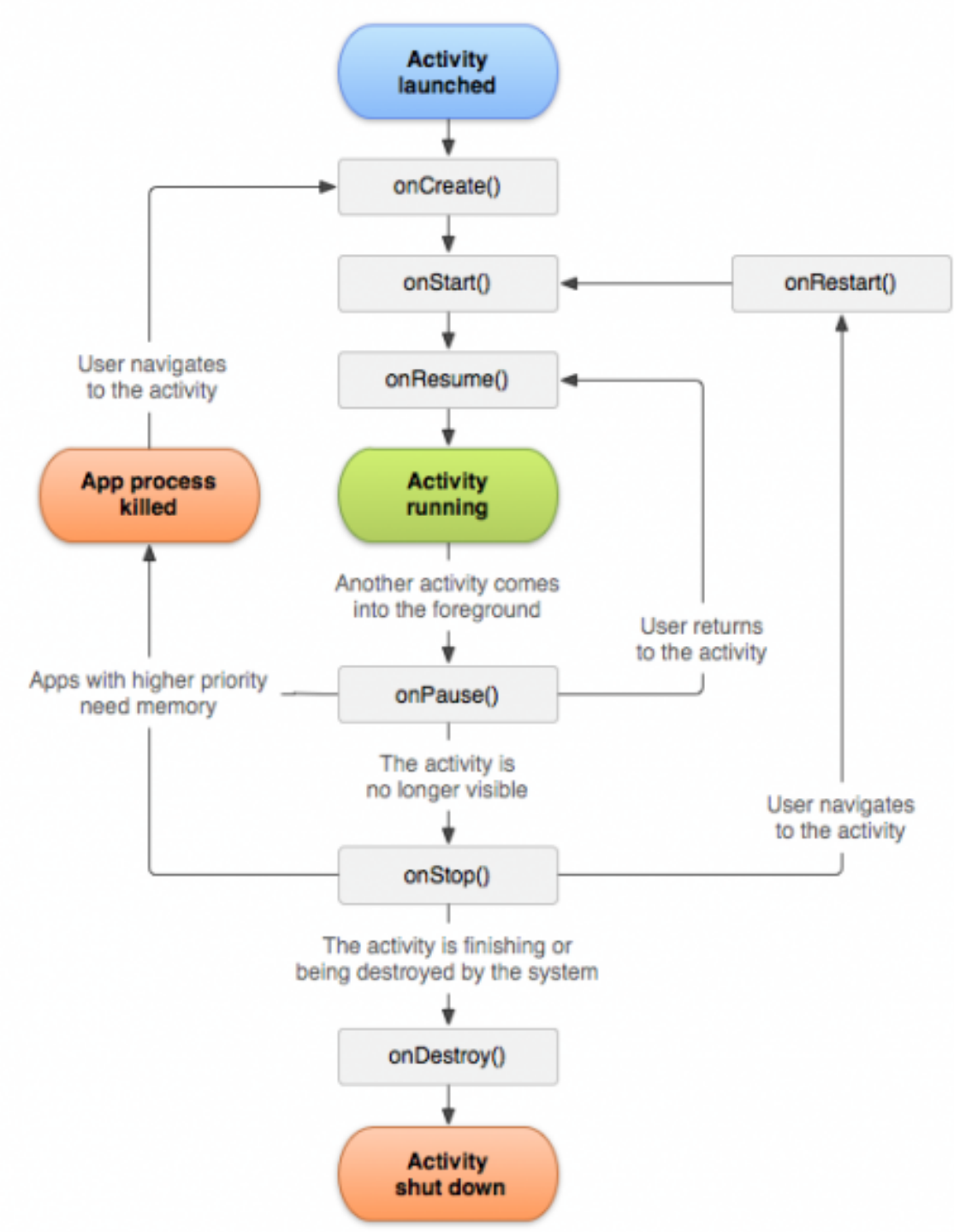
Med preklopom aktivnosti, se aktivnost, ki je imela trenutno nadzor nad zaslonom, neha izvajati in jo sistem potisne na t. i. zgodovinski sklad (ang. history stack). S tem se ohrani notranje stanje aktivnosti za kasnejše nadaljevanje izvajanja. Aktivnosti se lahko iz sklada tudi odstrani, in sicer v primeru nesmiselnega shranjevanja ali pomanjkanja pomnilnika. V povezavi s shranjevanjem na sklad, aktivnost med delovanjem prehaja skozi različna stanja. Pri tem govorimo o t. i. življenjskem ciklu aktivnosti, kar prikazuje slika 2.3.

Stanja aktivnosti

Aktivnost je v vsakem trenutku v enem izmed štirih možnih stanj:

- aktivna (ang. active) - aktivnost je prikazana na zaslonu v ospredju in se odziva na uporabnikove akcije, tj. dotike zaslona in pritiske tipk
- prekinjena (ang. paussed) - aktivnost se ne odziva na uporabnikove akcije, vendar je še vedno vsaj delno vidna na zaslonu. To se zgodi pri izbiri opcij ali pri potrjevanju dialogov
- ustavljena (ang. stopped) - aktivnost na zaslonu ni več vidna, a je še vedno na zgodovisnkem skladu
- uničena (ang. destroyed) - aktivnosti ni na zaslonu in je tudi na zgodovinskem skladu ni več

V sistemu je vedno aktivna le ena aktivnost, vse ostale pa so v enem od ostalih stanj.



Slika 2.3: Življenjski cikel aktivnosti.

Prehodi med stanji

Prehode med posameznimi stanji lahko zaznamo s pomočjo posebnih metod, ki jih definira razred Activity. Omenjene metode sistem avtomatsko kliče ob prehodih med stanji. Seznam metod je sledeč:

- `void onCreate(Bundle savedInstanceState)`
- `void onStart()`
- `void onResume()`
- `void onPause()`
- `void onStop()`
- `void onDestroy()`

Pri vsaki aktivnosti prazne implementacije teh metod že vsebuje razred Activity, obvezno je potrebno implementirati le metodo onCreate. Ta se kliče ob samem proženju aktivnosti. Če programer ne določi drugače, so prehodi med posameznimi stanji aktivnosti pri izvajanju aplikacije "nevidni", ob redefiniciji določenih metod pa lahko te prehode naredimo tudi vidne z izvedbo določenih akcij. Z možnostjo redefinicije metod imamo ob določenih dogodkih možnost odziva na spremembo. Pri redefiniciji metod je potrebno vedno najprej klicati metodo nadrazreda. Primer redefiniranja metode onResume:

```
protected void onResume() {
     super.onResume();
     games = getGamesThatMatchSavedFilter();
     showListView_games(games);
}
```

Metoda najprej kliče metodo nadrazreda, dobi tekme, ki ustrezajo nastavljenemu filtru (npr. za izbrano časovno obdobje), na koncu pa dobljene tekme še prikaže.

Prejemnik sporočil

Prejemnik sporočil je komponenta, ki se odziva na sporočila, ki jih dobi iz sistema ali drugih aplikacij. Primer sistemskega sporočila je npr. sporočilo, da je bil zaslon izklopljen ali da je baterija slaba. Primer sporočila od drugih aplikacij pa je npr. kadar določena aplikacija sporoči ostalim, da so bili določeni podatki preneseni in so na voljo za uporabo. Prejemnik sporočil ne vsebuje prikaza uporabniškega vmesnika, kljub temu pa lahko prikaže obvestilo v statusni vrstici in s tem opozori uporabnika, da se je zgodil nek dogodek. Običajno je prejemnik sporočil samo prehod do drugih komponent in opravlja minimalno delo. Kot primer lahko navedemo dogodek, ko prejemnik sporočil na podlagi dobljenega sporočila zažene določeno storitev.

2.1.4 Shranjevanje podatkov

Dostop do trajnega pomnilnika je na platformi Android v osnovi zelo omejen, saj lahko vsaka aplikacija privzeto dostopa samo do direktorija, ki ji je namenjen. Kljub temu stvar ni tako nemogoča, saj platforma nudi kar nekaj integriranih načinov dostopa do organiziranih podatkov, ki olajšajo shranjevanje. V grobem imamo pet možnih načinov dostopa:

- shranjevanje konfiguracije (ang. Properties)
- shranjevanje na internem pomnilniku (ang. Internal storage)
- shranjevanje na zunanjem pomnilniku (ang. External storage)
- SQLite podatkovna baza
- dostop do omrežja

Shranjevanje konfiguracije

Shranjevanje konfiguracije je najpreprostejši način hrambe, ki omogoča hrambo parov ključ-vrednost. V tem primeru ni potrebno dostopati do diska, saj za branje oz. pisanje poskrbi sistem sam. Za dostop do ustreznih objektov uporabimo eno izmed funkcij getPreferences ali getSharedPreferences.

Shranjevanje na internem pomnilniku

Pri shranjevanju na interni pomnilnik govorimo o direktoriju, ki je namenjen aplikaciji. Pomembne metode, ki se uporabljajo pri delu z direktorijem, so: openFileInput, openFileOutput, getFilesDir, getDir, deleteFile in fileList.

Shranjevanje na zunanjem pomnilniku

Dostop do datotek na zunanjem pomnilniku je podobno klasičnemu dostopu do datotečnega sistema v Javi. Edina posebnost je, da gre za zunanji pomnilnik (SD kartico), ki ni nujno vedno prisoten. Stanje zunanjega pomnilnika lahko zato v vsakem trenutku preverimo z metodo getExternalStorageState. Lokacijo zunanjega pomnilnika dobimo z uporabo getExternalFilesDir.

SQLite podatkovna baza

Na platformi Android imamo za kompleksnejše primere na voljo podporo za podatkovno bazo SQLite. Podatke lahko tako kot v drugih podatkovnih bazah organiziramo v tabele, poizvedbe nad njimi pa delamo s pomočjo poizvedovalnega jezika SQL.

Dostop do omrežja

Android aplikacija lahko poleg omenjenih načinov, uporablja tudi dostop do drugih računalnikov v omrežju. To se lahko naredi z uporabo standardnih Javinih razredov v paketu java.net.* ter razredov v paketu android.net.*. Pred

uporabo razredov v aplikaciji moramo sami aplikaciji odobriti pravice za dostop do mrežne povezljivosti.

2.2 Google Cloud Messaging - GCM

Preden začnemo opisovati storitev GCM, moramo poznati njen način, ki ga uporablja za pošiljanje podatkov do določene mobilne naprave. Obstajata dva načina - Pull in Push način. GCM storitev uporablja slednjega.

Pull tehnologija

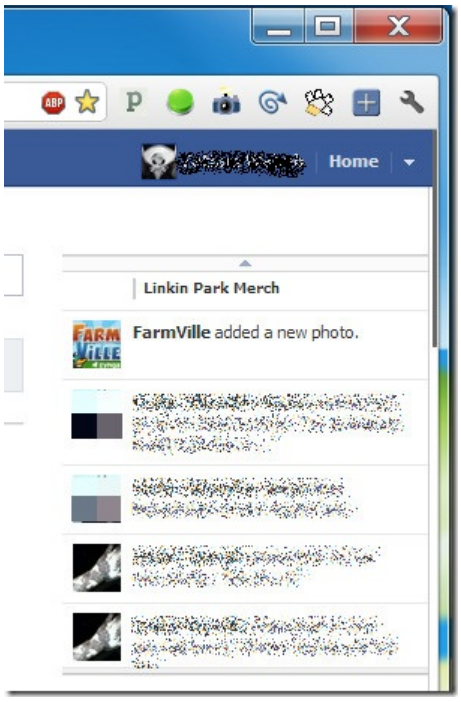
Pull tehnologija je način internetne komunikacije, kjer prvotni zahtevek (ang. initial request) za podatke poda odjemalec (ang. client), na zahtevo pa se odzove strežnik (ang. server). Ta odjemalcu pošlje podatke, ki ga zanimajo.

Pull način je učinkovit v primeru, kadar spletno mesto ne pozna uporabnikov, ki ga obiskujejo. Npr. uporabniki iščejo nekaj specifičnega, spletno mesto pa ne more vedeti v naprej, kaj je to. Pull način je tako široko uporabljen na internetu pri HTTP zahtevkih spletnih strani.

Večina spletnih virov (ang. web feeds) npr. RSS, uporabljajo pull tehnologijo. Uporabnikov RSS bralnik (ang. RSS reader) periodično sprašuje strežnik, ali je na voljo kaj nove vsebine. V primeru da je, strežnik pošlje novo vsebino. Nikoli se ne zgodi, da bi strežnik poslal vsebino odjemalcu brez predhodnega zahtevka za to. Neprestano spraševanje po novi vsebini je neučinkovito in seveda obremenjuje povezavo, zato se je razvila druga tehnologija - PUSH.

Push tehnologija

Push tehnologija je način internetne komunikacije, kjer zahtevek za transakcijo ne pride od odjemalca (tako kot pri PULL tehnologiji), ampak od strežnika oziroma od tistega, ki je strežniku poslal nove podatke. Če strežnik dobi nove



Slika 2.4: Facebook - primer PUSH tehnologije, ko na desni strani zaslona dobivamo nova obvestila brez osveževanja strani.

podatke, jih pošlje naprej odjemalcu oziroma več odjemalcem. Tako jim ni potrebno periodično spraševati strežnika po novih podatkih.

Tipični primer push načina komunikacije je neposredno sporočanje (ang. instant messaging). Najbolj znan primer je Skype. Besedilna sporočila in tudi datoteke so potisnjene (ang. pushed) k uporabniku takoj, ko so na voljo (tj. ko uporabnik na drugi strani napiše sporočilo in pritisne tipko Enter). Znan primer je tudi Facebook - uporabniki prejmejo nova obvestila brez osveževanja strani (glej sliko 2.4).

Push tehnologija je našla svoje mesto tudi v mobilnih napravah. Google je na primer za mobilne naprave z operacijskim sistemom Android zagnal storitev Google Cloud Messaging (GCM).

2.2.1 GCM storitev

Google Cloud Messaging (GCM) je storitev, ki omogoča pošiljanje podatkov oziroma sporočil iz našega lastnega strežnika vsem uporabnikom Android naprav. Za uporabljanje GCM storitve v osnovi potrebujemo:

- aplikacijski strežnik
- mobilno napravo z operacijskim sistemom Android

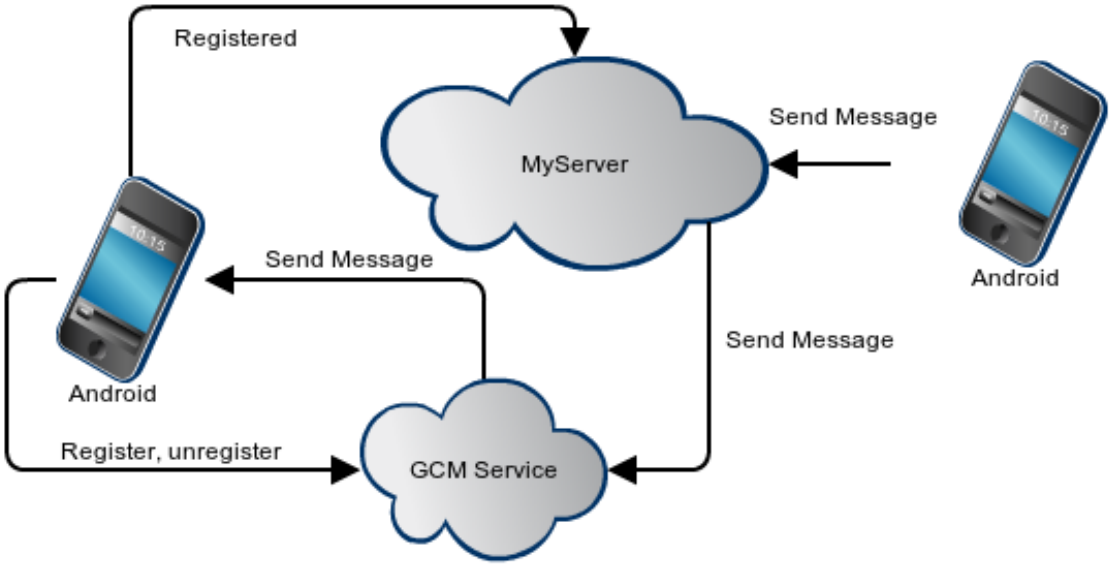
Osnovno delovanje GCM storitve ter pošiljanje sporočila prikazuje slika 2.5.

Poznamo dve vrsti sporočila, ki se od aplikacijskega strežnika do Android naprave pošlje preko GCM strežnikov:

- sporočilo vsebuje le podatek, da so na strežniku novi podatki (npr. prijatelj je na strežnik naložil nov video)
- sporočilo že vsebuje nove podatke (v tem primeru lahko pošljemo največ 4 kb podatkov naenkrat)

Glavne značilnosti GCM-ja:

- vsakemu strežniku je dovoljeno poslati sporočilo aplikaciji, ki teče na mobilni napravi z operacijskim sistemom Android
- mobilna naprava lahko prejme sporočilo tudi, ko aplikacija, ki podpira sprejemanje sporočil, ne teče. V takem primeru (na mobilno napravo prispe sporočilo, aplikacija ne teče) sistem sam zbudi aplikacijo, aplikacija pa nato poskrbi za prikaz sporočila.
- nima vgrajenih načinov za obdelavo podatkov. GCM preprosto vse podatke, ki jih dobi, v enaki obliki pošlje naprej aplikaciji. Aplikacija mora nato sama poskrbeti za obdelavo oz. prikaz podatkov. Na primer:



Slika 2.5: GCM storitev - osnovno delovanje.

  - prikaže sporočilo (angleško: notification) le na vrhu zaslona v statusni vrstici (angleško: status bar)
  - prikaže podatke na celotnem zaslonu
  - v ozadju poskrbi za sinhronizacijo podatkov

- za delovanje potrebuje mobilno napravo, na kateri teče operacijski sistem Android različice 2.2 ali novejši ter ima nameščeno aplikacijo Google Play ali pa emulator, na kateremu prav tako teče vsaj Android različice 2.2.

- uporablja obstoječo povezavo z Google storitvami. Za naprave z operacijskim sistemom različice starejše od 4.0.4, morajo uporabniki na mobilnih napravah nastaviti Google račun, medtem ko Google račun na napravah z operacijskim sistemom od različice 4.0.4 naprej ni več potreben.

2.2.2 Arhitekturni pregled GCM storitve

V tem poglavju si bomo pogledali način delovanjaGCM storitve. Tabela 2.2 povzema ključne pojme in koncepte vključene v GCM. Tabela je razdeljena v dve kategoriji:

- komponente (ang. Components) - fizične entitete, ki so potrebne pri uporabi GCM
- poverilnice (ang. Credentials) - enolični identifikatorji (ang. ID) in žetoni (ang. token) uporabljeni v različnih fazah pošiljanja/prejemanja sporočil, ki zagotovijo, da so vsi sodelujoči avtenticirani in da poslano sporočilo doseže prave mobilne naprave.

Komponente	
Mobilna naprava	Naprava, ki poganja operacijski sistem Android in ima nameščeno aplikacijo, ki uporablja GCM. Operacijski sistem mora biti različice 2.2 ali novejši, na napravi pa mora biti nameščena aplikacija Google Play Store. Prav tako morajo imeti naprave z operacijskim sistemom starejšim od 4.0.4 vsaj en prijavljen Google račun. Alternativno lahko za potrebe testiranja uporabimo tudi emulator z Androidom 2.2 in z Google API-jem.
Aplikacijski strežnik	Aplikacijski strežnik, ki ga razvijalec postavi kot del implementacije GCM za njegovo aplikacijo. Strežnik pošilja podatke na naprave z Androidom preko GCM strežnika.
GCM strežniki	Googlovi strežniki, ki od aplikacijskega strežnika prejmejo sporočilo in ga pošljejo naprej na mobilne naprave.

Poverilnice	
Pošiljateljev enolični identifikator (ang. Sender ID)	Številka projekta pridobljena iz API konzole. Številka je uporabljena v registracijskem procesu za identifikacijo aplikacije, ki ji je dovoljeno pošiljati sporočila na druge naprave.
Aplikacijski enolični identifikator (ang. Application ID)	Aplikacija, ki se registrira za prejemanje sporočil. Aplikacija je identificirana preko imena paketa (ang. package name) iz manifesta. To zagotavlja, da so sporočila poslana samo pravim aplikacijam.
Registracijski enolični identifikator (ang. Registration ID)	Enolični identifikator (ang. ID) izdan aplikaciji od GCM strežnikov. Ta aplikaciji dovoljuje prejemanje sporočil. Po prejemu registracijskega enoličnega identifikatorja, ga aplikacija pošlje aplikacijskemu strežniku, ki s tem ve, da je mobilna naprava registrirana za prejemanje sporočil preko določene aplikacije. Z drugimi besedami, registracijski enolični identifikator je vezan na točno določeno aplikacijo, ki teče na točno določeni napravi.
Google račun uporabnika	Za delovanje GCM mora imeti mobilna naprava vsaj en Google račun, če na napravi tečejo Android različice starejše od 4.0.4.
Pošiljateljev avtentikacijski žeton (ang. Sender Auth Token)	API ključ, ki je shranjen na aplikacijskem strežniku, omogoča avtenticiran dostop do Googlovih storitev. API ključ je pri sporočilih vključen v glavi POST zahteve.

Tabela 2.2: Ključni pojmi in koncepti vključeni v GCM.

Življenjski cikel GCM storitve

GCM storitev ima naslednji življenjski cikel:

- omogočanje GCM storitve. Aplikacija, ki teče na mobilni napravi z Androidom se registrira za prejemanje sporočil.
- Pošiljanje sporočila. Aplikacijski strežnik pošlje sporočilo na mobilno napravo preko GCM strežnika.
- Prejemanje sporočila. Aplikacija na mobilni napravi prejme sporočilo od GCM strežnika.

Našteti procesi so bolj natančno opisani v nadaljevanju.

GCM - Omogočanje GCM storitve

Omogočanje GCM storitve je zaporedje dogodkov, ki se zgodijo, ko se aplikacija na mobilni napravi z Androidom registrira za prejemanje sporočil:

1. prvič, ko hoče aplikacija uporabiti GCM storitev, pošlje registracijsko zahtevo GCM strežniku. Registracijska zahteva vsebuje pošiljateljev enolični identifikator ter aplikacijski enolični identifikator.
2. V primeru uspele registracije GCM strežnik pošlje aplikaciji na mobilni napravi registracijski enolični identifikator. Aplikacija naj bi bila napisana tako, da dobljen podatek shrani za kasnejšo uporabo. Z njim lahko aplikacija ob vsakem zagonu preveri, ali je že registrirana. Opozorilo: Google lahko občasno spreminja registracijski enolični identifikator, tako da mora aplikacija znati pridobiti nov registracijski enolični identifikator, če je to potrebno.
3. Za dokončanje registracije, mora aplikacija na mobilni napravi aplikacijskemu strežniku poslati registracijski enolični identifikator. Strežnik ta podatek po navadi shrani v podatkovno bazo.

Registracijski enolični identifikator je v uporabi dokler se aplikacija ne odjavi od storitve GCM oziroma dokler Google ne osveži tega podatka.

GCM - Pošiljanje sporočila

Da lahko aplikacijski strežnik pošlje sporočilo aplikaciji na mobilno napravo z Androidom, morajo biti izpolnjene naslednje zahteve:

- aplikacija mora imeti pridobljen registracijski enolični identifikator, ki ji dovoljuje prejemanje sporočil na točno določeno mobilno napravo
- aplikacijski strežnik je shranil registracijski enolični identifikator, ki mu ga je poslala aplikacija
- aplikacijski strežnik mora imeti nastavljen API ključ.

Dogodki, ki se zgodijo, ko aplikacijski strežnik pošlje sporočilo:

1. aplikacijski strežnik pošlje sporočilo GCM strežniku
2. Google da sporočilo v čakalno vrsto ter ga shrani za primer, če mobilna naprava trenutno ni dosegljiva
3. ko je naprava dosegljiva, ji Google pošlje sporočilo.

GCM - Prejemanje sporočila

Dogodki, ki se zgodijo, ko aplikacija, nameščena na mobilni napravi z Androidom, prejme sporočilo:

1. operacijski sistem prejme dohodno sporočilo in iz njega razbere pare ključ/vrednost (ang. key/value pairs), če so le ti prisotni.
2. operacijski sistem poda pare ključ/vrednost ciljni aplikaciji
3. aplikacija pridobi neobdelane podatke (vrednosti) preko ključev in jih obdela.

Namestitev aplikacije, ki vključuje GCM storitev

Kadar želi uporabnik namestiti aplikacijo, ki vključuje GCM storitev, na svoj telefon z operacijskim sistemom Android, ga trgovina Google Play Store obvesti, da aplikacija vključuje GCM storitev. Uporabnik se mora s tem strinjati pred dokončno namestitvijo aplikacije.

2.2.3 Vloga aplikacijskega strežnika

Aplikacija, ki vključuje GCM storitev, bi bila neuporabna, če ne bi imeli aplikacijskega strežnika, ki zna:

- komunicirati z odjemalcem
- pošiljati HTTPS zahteve GCM strežniku
- obravnavati zahteve in jih ponovno pošiljati po potrebi
- shraniti API ključ in odjemalčeve registracijske enolične identifikatorje. API ključ je pri sporočilih vključen v glavi POST zahteve.

Aplikacijski strežnik - Pošiljanje sporočil

V tem poglavju bomo opisali, kako aplikacijski strežnik pošlje sporočilo mobilnim napravam. Pomembno je naslednje:

- aplikacijski strežnik lahko podatke pošlje eni ali pa več mobilnim napravam. Sporočilu, ki je poslano več napravam hkrati, pravimo multicast sporočilo.
- poslani in prejeti podatki so lahko v dveh oblikah: navaden tekst ali JSON struktura
- pri pošiljanju multicast sporočil moramo obvezno uporabiti JSON strukturo.

Preden lahko aplikacijski strežnik pošlje sporočilo aplikaciji na mobilno napravo z Androidom, mora od aplikacije prejeti registracijski enolični identifikator.

Oblika zahteve

Za pošiljanje sporočila mora aplikacijski strežnik poslati POST zahtevo na naslov: https://android.googleapis.com/gcm/send

Zahteva je sestavljena iz dveh delov: HTTP glava (ang. header) in HTTP telesa (ang. body). HTTP glava mora vsebovati:

- avtorizacijo: ključ=API_KLJUČ
- tip vsebine:
  - application/json za JSON strukturo
  - application/x-www-form-urlencoded;charset=UTF-8 za navaden tekst

Primer:

```
Content-Type:application/json
Authorization:key=AIzaSyB-1uEai2WiUapxCs2Q0GZYzPu7Udno5aA
{
     "registration_ids" : ["APA91bHun4MxP5egoKMwt2KZFBaFUH-
      1RYqx..."],
     "data" : {
      ...
      },
}
```

Opozorilo: če tip vsebine v http glavi ni nastavljen, se privzame, da je v sporočilu vsebovan navaden tekst.

Vsebina HTTP telesa je odvisna ali uporabljamo JSON strukturo ali navaden tekst. Za JSON strukturo mora vsebovati niz znakov (ang. string), ki predstavljajo JSON objekt z naslednjimi polji:

- registration_ids (obezno polje)
- collapse_key (neobvezno polje)
- data (neobvezno polje)
- delay_while_idle (neobvezno polje)
- time_to_live (neobvezno polje)
- restricted_package_name (neobvezno polje)
- dry_run (neobvezno polje)

V primeru, da uporabljamo navaden tekst namesto JSON strukture, moramo zgoraj omenjena polja nastaviti kot http parametre poslane v telesu, sintaksa pa je malenkost drugačna. Obvezno polje je še vedno samo registration_id.

Primeri:

- Najmanjša možna zahteva z uporabo JSON strukture

```
{ "registration_ids": [ "42" ] }
```

- Ista zahteva z uporabo navadnega teksta

```
registration_id=42
```

- Sporočilo z vsemi možnimi polji in s šestimi prejemniki z uporabo JSON strukture

```
{
     "collapse_key": "score_update",
     "time_to_live": 108,
     "delay_while_idle": true,
     "data": {
          "score": "4x8",
          "time": "15:16.2342"
     },
     "registration_ids":["4", "8", "15", "16", "23", "42"]
}
```

- Isto sporočilo z uporabo navadnega teksta, ampak le enim prejemnikom (več kot en prejemnik ni možen)

```
collapse_key=score_update&time_to_live=108&
delay_while_idle=1&data.score=4x8&
data.time=15:16.2342&registration_id=42
```

Oblika odziva

Pri pošiljanju sporočila lahko dobimo dva možna odziva:

- sporočilo je bilo uspešno obdelano
- GCM strežnik je zavrnil zahtevo.

Pri uspešno obdelanem sporočilu dobimo HTTP odziv s statusom 200, pri zavrnitvi zahteve pa dobimo HTTP odziv s statusom različnim od 200 (možni statusi so npr.: 400, 401, 503). V tabeli 2.3 so statusi HTTP odziva predstavljeni bolj podrobno.

Odziv	Opis
200	Sporočilo je bilo uspešno obdelano. Telo odziva vsebuje več informacij o statusa sporočila, format pa je odvisen ali uporabljamo JSON strukturo ali navaden tekst.
400	Velja samo za sporočila v JSON strukturi. Status 400 nam pove, da zahteve ni bilo mogoče obdelati kot JSON strukturo, ali pa zahteva vsebuje nedovoljena polja (npr. v neko polje zapišemo niz znakov namesto pričakovane številčne vrednosti).
401	Pojavil se je problem pri avtentikaciji pošiljateljevega računa.
5xx	Napake v razponu od 500-599 (npr. 500 ali 503) nam povejo, da se je med obdelovanjem zahteve bodisi zgodila napaka na GCM strežniku bodisi je GCM strežnik začasno nedosegljiv.

Tabela 2.3: Možni statusi HTTP odziva pri pošiljanju sporočila.

Pri uspešni JSON zahtevi (HTTP status je 200), telo odziva vsebuje JSON objekt s polji, navedenimi v tabeli 2.4.

Polje	Opis
multicast_id	Unikaten enolični identifikator (številka), ki identificira multicast sporočilo
success	Število sporočil, ki so bila obdelana brez napak
failure	Število sporočil, ki niso bila uspešno obdelana
canonical_ids	Število rezultatov, ki vsebujejo kanonični registracijski enolični identificator (ang. canonical registration ID)
results	Polje objektov, ki predstavljajo status obdelanih sporočil

Tabela 2.4: Polja prisotna v telesu pozitivnega odziva po JSON zahtevi.

Primera:

- JSON sporočilo uspešno poslano enemu naslovniku

```
{
     "multicast_id": 108,
     "success": 1,
     "failure": 0,
     "canonical_ids": 0,
     "results": [
          { "message_id": "1:08" }
      ]
}
```

- Odgovor na isto sporočilo v navadnem tekstu

```
id=1:08
```

2.3 PHP

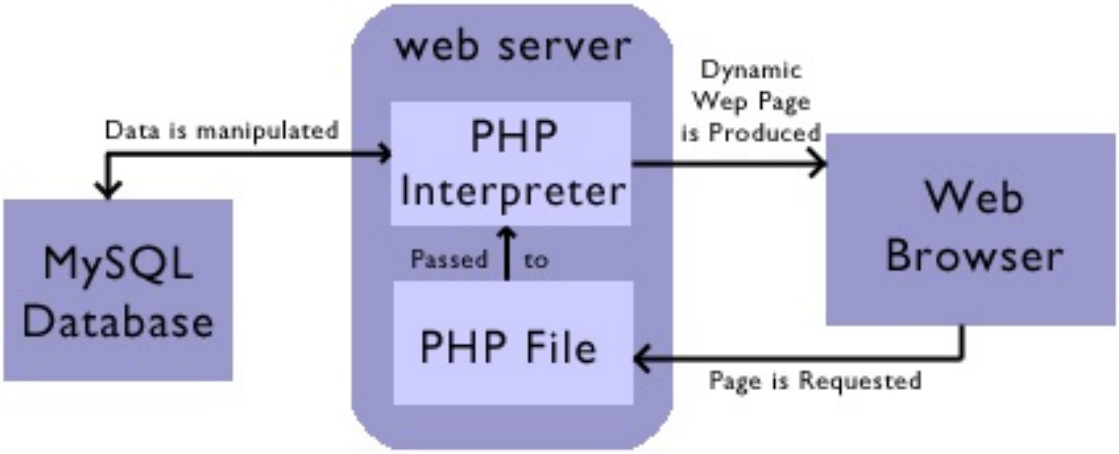
PHP je razširjen odprtokodni strežniški skriptni jezik, k ise uproablja za razvoj dinamičnih spletnih strani. Kratica PHP trenutno stoji za PHP Hypertext Preprocessor, na začetku (v letu 1995) pa je kratica PHP pomenila Personal Home Page Tools (PHP Tools). PHP je razvil dansko-kanadski programer Rasmus Lerdorf, da bi na spletni strani prikazal svoj življenjepis in hkrati zajemal podatke obiskovalcev strani. Kasneje je funkcionalnosti razširil do te mere, da se je jezik lahko uporabljal za razvoj dinamičnih spletnih strani. Z verzijo 5.0, ki je bila izdana leta 2005, je PHP postal objektno orientiran jezik.

PHP primarno teče na spletnem strežniku, ki PHP izvorno kodo jemlje za vhod in generira spletno stran kot izhod. Poleg tega PHP omogoča tudi zaganjanje skript v ukaznem načinu in kreiranje grafičnih aplikacij. Slika 2.6 prikazuje proces generiranja spletne strani. Koraki so naslednji:

- brskalnik (odjemalec) poda zahtevo strežniku
- interpreter združi podatke iz baze podatkov in datotek v HTML dokument
- HTML dokument se pošlje odjemalcu.

2.4 MySQL

MySQL (My Structured Query Language) je odprtokodni sistem za upravljanje s podatkovnimi bazami in implementira relacijsko podatkovno bazo, ki za delo s podatki uporablja poizvedovalni jezik SQL (Structured Query Language). MySQL deluje po principu odjemalec-strežnik, pri čemer lahko strežnik namestimo kot sistem, porazdeljen na več strežnikov. Obstaja veliko število odjemalcev, zbirk ukazov in programskih vmesnikov za dostop do podatkovne baze MySQL. Pri povezovanju s podatkovnim strežnikom moramo podati podatke kot so:



Slika 2.6: PHP - osnovno delovanje.

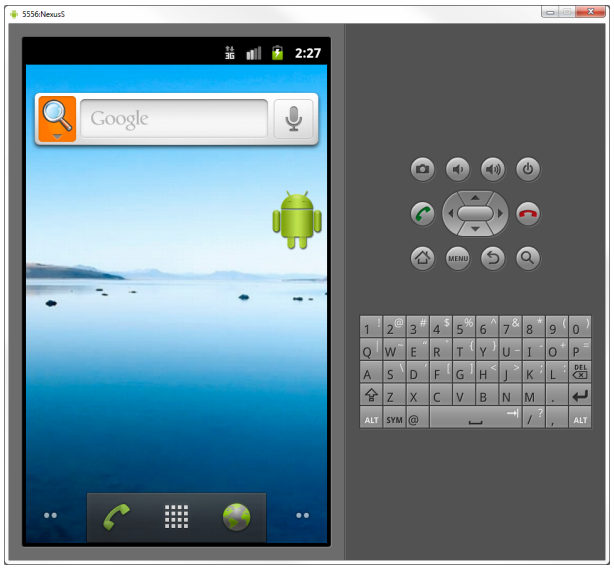
- uporabniško ime
- geslo
- ime podatkovne baze
- gostitelj (ang. host).

Trenutno je MySQL najbolj popularna podatkovna baza za uporabo v spletnih aplikacijah.

2.5 Razvojna orodja

2.5.1 Eclipse

Eclipse je odprtokodno programsko-razvojno okolje, ki podpira razvoj v več programskih jezikih. Med njimi je seveda tudi Java, programski jezik, v katerem se primarno razvija aplikacije za Android. Pri razvoju aplikacij za platformo Android je poleg Eclipsa potrebno namestiti Android SDK, v Eclipse pa namestiti vtičnik ADT (Android Development Tools). Po namestitvi vtičnika



Slika 2.7: Android emulator na operacijskem sistemu Windows.

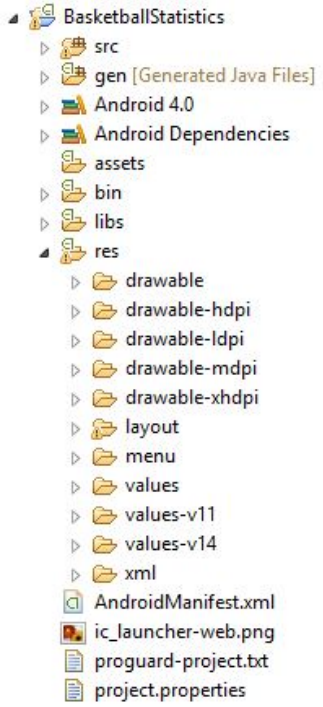
lahko začnemo uporabljati emulator, s pomočjo katerega preizkusimo delovanje aplikacije, ki jo razvijamo, kar na računalniku. Primer emulatorja prikazuje slika 2.7.

Struktura projekta v Eclipsu pri razvoju Android aplikacije

Preden začnemo s pisanjem Android aplikacije v Eclipsu, moramo poznati strukturo projekta.Vedeti je potrebno, kje se definira izgled aplikacije (tj. grafični vmesnik), in kje logiko, kako se bo aplikacija obnašala. Slika 2.8 prikazuje strukturo projekta v Eclipsu.

Struktura projekta - programska koda in knjižnice

Programska koda je, kot že rečeno, napisana v programskem jeziku Java in se nahaja v mapi src. Za razširitev funkcionalnosti lahko poleg svoje kode v projekt vključimo oz. dodamo tudi knjižnice. Knjižnic ne damo v isto mapo kot kodo, ampak v za to namenjeno mapo libs.



Slika 2.8: Struktura projekta v razvojnem orodju Eclipse pri razvoju Android aplikacije.

Struktura projekta - izgled aplikacije

Izgled aplikacije pri razvoju za Android platformo definiramo v XML (ang. Extensible Markup Language) datotekah, ki jih najdemo v mapi res. Z izgledom aplikacije ni mišljena le postavitev posameznih elementov na zaslonu, ampak tudi njihova oblika, barva in napis na njih. Mapa res je nadalje razdeljena na več podmap, glavne pa so:

- XML datoteke za definiranje pozicije elementov na zaslonu se nahajajo v mapi Layout

- za grafične elemente, ki jih vključujemo v izgled aplikacije, je v projektu namenjena mapa drawable. Ker operacijski sistem Android teče na mobilnih napravah z zasloni, ki imajo različno gostoto točk (ang. Screen

density), imamo več map drawable. Mape imajo imena oblikovana na način drawable-ime_gostote_točk (npr. drawable-hdpi, drawable-mdpi)

- posamezne barve lahko definiramo v XML datoteki color.xml, posamezne napise pa v strings.xml, znotraj mape values.

2.5.2 Android SDK

Android SDK zagotavlja API knjižnice in razvojna orodja, ki so potrebna za razvoj, testiranje in razhroščevanje (ang. debugging) Android aplikacij. Vsebuje torej razhroščevalnik, knjižnice, emulator, dokumentacijo, primere in vodiče.

Z vsako novo različico operacijskega sistema Android, Google izda tudi novo različico SDK. Novejše različice vsebujejo knjižnice z razredi in funkcijami, ki jih prejšnje različice niso imele. Pri razvoju Android aplikacije se moramo zato pred začetkom razvijanja aplikacije odločiti, kateri SDK bomo uporabljali. S tem v splošnem povemo, koliko stare naprave bodo še lahko poganjale razvito aplikacijo.

Za razvoj naše aplikacije smo uporabili SDK različico, ki podpira API nivo 16 (uporablja ga operacijski sistem Android 4.1), vendar smo zagotovili združljivost tudi za nazaj, vse do API nivoja 8 (uporablja ga operacijski sistem Android 2.2). Omenjene podatke, kateri operacijski sistem uporablja kateri API, lahko vidimo v tabeli 2.1.

Poglavje 3

Mobilna aplikacija

Mobilna aplikacija je glavni del celotnega sistema, ki omogoča pisanje statistike na košarkarskih tekmah in spremljanje rezultatov drugih tekem, če to želimo. Aplikacija namreč omogoča, da lahko tisti, ki piše statistiko za neko tekmo, deli rezultat tekme z vsemi, ki imajo nameščeno to aplikacijo.

Pri razvoju mobilne aplikacije smo največ časa posvetili pošiljanju sporočil našemu strežniku, ki je predstavljen v naslednjem poglavju in prejemanju sporočil od GCM strežnika. Našem strežniku aplikacija pošlje sporočilo, ki ni nič drugega kot trenutni rezultat tekme. Naš strežnik nato pošlje sporočilo GCM strežniku, ki pošlje prejeto sporočilo vsem, ki imajo nameščeno aplikacijo in so prijavljeni za prejemanje sporočil, tj. trenutnega rezultata določene tekme.

3.1 Načrtovanje in razvoj mobilne aplikacije

Načrtovanje in razvoj mobilne aplikacije smo razdelili na več smiselnih manjših delov. Pomembnejši deli so:

- načrtovanje podatkovne baze
- razvoj aktivnosti za pisanje statistike ter pošiljanje sporočil strežniku

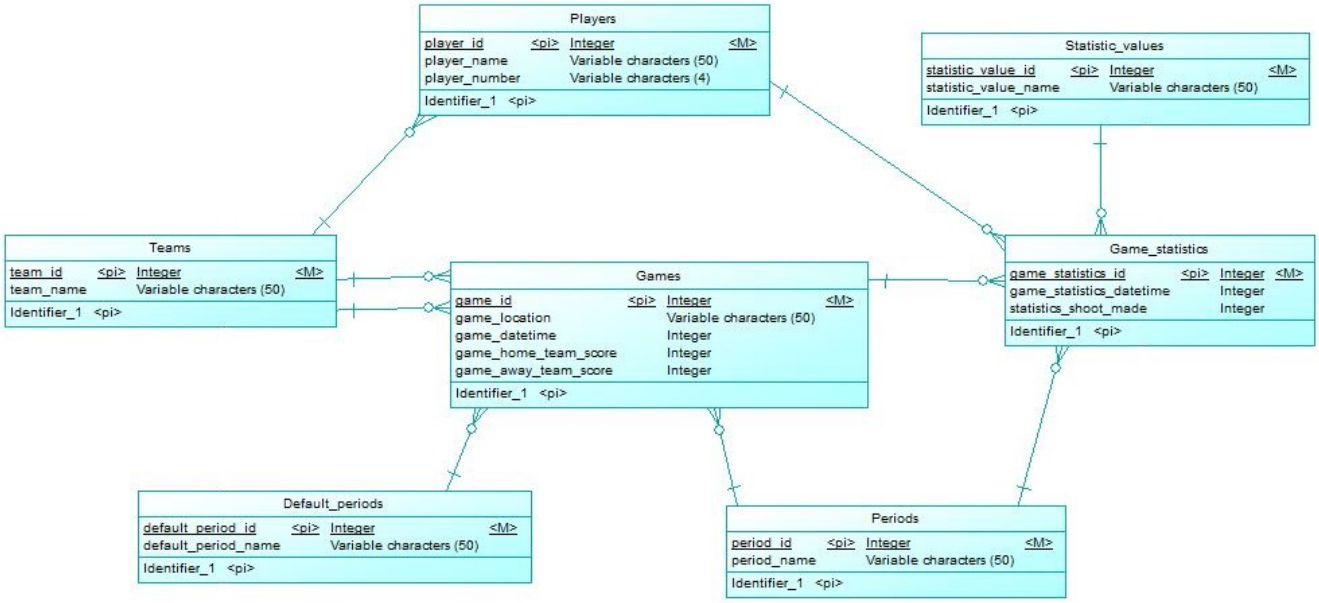
- razvoj aktivnosti za prejemanje sporočil od GCM strežnika.

Poleg omenjenih delov je bilo potrebno razviti tudi druge aktivnosti, brez katerih aplikacija ne bi mogla delovati. Sem spadajo:

- aktivnosti za urejanje moštev in igralcev (statistike za posamezno tekmo ne moremo pisati, če ne vemo, kateremu igralcu naj pripišemo določen statističen element, npr. izgubljeno žogo ali zadet prosti met)
- aktivnost z nastavitvami
- aktivnost za prikazovanje tekem (vse tekme, za katere smo pisali statistiko, se shranjujejo v podatkovno bazo - tako lahko brez problema pogledamo statistiko za tekmo, ki se je odvijala eno leto nazaj) - to je vstopna aktivnost aplikacije, ki jo lahko vidimo na sliki 3.1.

Slika 3.1: Vstopna aktivnost aplikacije.

V nadaljevanju so opisani pomembnejši deli mobilne aplikacije.



Slika 3.2: Model podatkovne baze za aplikacijo.

3.1.1 Načrtovanje podatkovne baze

V mobilni aplikaciji lahko podatke shranjujemo na več načinov. V našem primeru smo izbrali podatkovno bazo SQLite. Pred pričetkom programiranja aplikacije, smo izdelali model podatkovne baze, ki ga prikazuje slika 3.2.

V nadaljevanju sledi opis pomembnejših entitet.

Games

Tabela Games vsebuje podatke o vseh tekmah, za katere je uporabnik aplikacije pisal statistiko in vsebuje naslednje atribute:

- game_id - umetni enolični identifikator
- game_location - lokacija igranja tekme
- game_datetime - datum in čas igranja tekme

- game_home_team_score - število točk, ki ga je dosegla domača ekipa
- game_away_team_score - število točk, ki ga je dosegla gostujoča ekipa

Poleg omenjenih atributov so v tabeli še tuji ključi:

- home_team_id - domača ekipa
- away_team_id - gostujoča ekipa
- default_period_id - podatek, ki pove ali ima tekma četrtine ali polovice (po navadi se igrajo tekme 4 krat po 10 minut ali pa 2 krat po 20 minut)
- current_period_id - podatek, ki pove katera četrtina/polovica trenutno poteka na tekmi

Game_statistics

Tabela Game_statistics vsebuje podatke o statistiki za določeno tekmo. Vsebuje vse dogodke (npr. zadet koš, izgubljena žoga) v točno takem vrstnem redu, kot so se zgodili na tekmi. Tabela ima naslednje atribute:

- game_statistics_id - umetni enolični identifikator
- game_statistics_datetime - datum in čas vnosa posameznega statističnega elementa (npr. dosežen koš, izgubljena žoga)
- statistics_shoot_made - podatek, ki pove, ali je igralec, ki je metal na koš, zadel ali zgrešil

Poleg omenjenih atributov so v tabeli še tuji ključi:

- game_id - pove nam, za katero tekmo vpisujemo podatke

- statistic_value_id - pove nam, kateri statistični element smo pripisali posameznemu igralcu
- player_id - igralec, kateremu smo pripisali nek statistični element
- period_id - podatek, ki pove v kateri četrtini/polovici je bil statistični element zapisan

Teams

Tabela Teams vsebuje imena ekip, ki jih je uporabnik dodal v aplikaciji. Tabela ima naslednja atributa:

- team_id - umetni enolični identifikator
- team_name - ime ekipe

Players

Tabela Players vsebuje podatke o igralcih, ki jih je uporabnik vnesel v aplikacijo. Vsak igralec lahko pripada zgolj eni ekipi. Tabela vsebuje naslednje atribute:

- player_id - umetni enolični identifikator
- player_name - ime igralca
- player_number - številka igralca

Poleg omenjenih atributov je v tabeli še tuji ključ:

- team_id - ekipa, za katero igralec igra

3.1.2 Razvoj aktivnosti za pisanje statistike ter pošiljanje sporočil strežniku

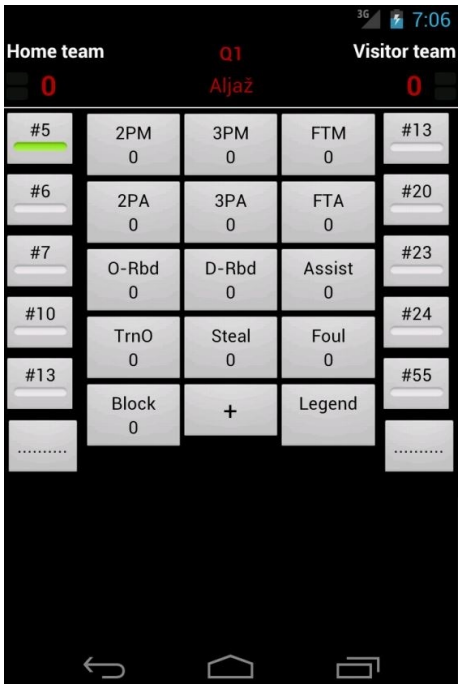
Razvoj aktivnosti za pisanje statistike in pošiljanje sporočil strežniku smo razdelili na dva dela. Prvi del je bil pisanje statistike, drugi del pa pošiljanje sporočil strežniku.

Pisanje statistike

Na mobilnih napravah oziroma mobilnih telefonih je zaslon relativno majhen,na katerega smo morali razporediti vse potrebne elemente za pisanje statistike. Poleg imena domače in gostujoče ekipe, trenutnega rezultata, trenutne četrtine/polovice, podatka ali je ekipa v bonusu ali ne in imena trenutno izbranega igralca, je bilo na zaslon potrebno dodati še vse statistične elemente, ki jih lahko posameznemu igralcu pripišemo. Seveda mora biti na zaslonu vidna tudi trenutna postava obeh ekip in gumb, ki omogoči spremembo postave. Končno postavitev elementov na zaslonu lahko vidimo na sliki 3.3.

Uporabniški vmesnik smo napisali v XML jeziku, saj omogoča enostavno postavitev elementov na zaslonu. Uporabili smo elementa LinearLayout in RelativeLayout, ki omogočata pravilen izris elementov na mnogih zaslonih, ne glede na njihovo ločljivost. Zgornji del zaslona vsebuje splošne podatke, vsebuje pa 4 elemente ImageView za prikaz bonusa obeh ekip ter šest elementov TextView za:

- prikaz imena domače ekipe
- prikaz imena gostujoče ekipe
- prikaz točk domače ekipe
- prikaz točk gostujoče ekipe
- prikaz imena trenutno izbranega igralca



Slika 3.3: Aktivnost za pisanje statistike.

- prikaz trenutne četrtine/polovice

Pod splošnimi podatki najdemo statistične elemente na sredini, na vsaki strani pa imamo po pet igralcev vsake ekipe, ki so trenutno v igri. Na vsaki strani je pod igralci še gumb za zamenjavo igralcev. Omenjeni gumb je element tipa Button, igralce pa predstavljajo elementi tipa ToggleButton. V splošnem se ta tip gumba uporablja za nastavitve, ki so lahko vklopljene ali izklopljene. Pri tem smo izkoristili lastnost, da v primeru vklopljenosti sveti lučka. Na sliki 3.3 je tako izbran igralec domače ekipe številka pet. Vidimo tudi, da je igralcu ime Aljaž, kar je izpisano v zgornjem delu zaslona.

Za statistične elemente, čeprav na prvi pogled temu tako, nismo uporabili elementov tipa Button. Potrebovali smo namreč gumb z dvema vrsticama. Zgornja vrstica v posameznem gumbu predstavlja ime statističnega elementa, spodnja pa števec. Ta nam pove, koliko točk, skokov, asistenc, itd. je izbrani

igralec zbral do tega trenutka. Posamični gumb, ki predstavlja statistični element, smo naredili na naslednji način:

- za izgled gumba smo uporabili RelativeLayout, ki smo mu določili:
  - clickable="true" - s tem povemo, da se na element da klikniti, kar pomeni, da deluje na enak način kot element Button
  - background="@android:drawable/btn_default" - s tem povemo, da naj ima element enak izgled kot element Button
- ime statističnega elementa in števec sta TextView, obema pa smo določili:
  - textAppearance="?android:attr/textAppearanceButton" - s tem povemo, da naj tekst izgleda tako, kot je to primer pri elementu Button

Statističnim elementom smo dodali še gumb, s katerim je možno brisanje dogodkov. Na sliki 3.3 je to gumb z znakom plus. Dokler je na gumbu izpisan znak plus to pomeni, da s klikom na gumb nekega statističnega elementa, povečamo števec. Ob kliku na gumb z znakom plus, se ta spremeni v gumb z znakom minusa. V tem primeru lahko števec za izbrani statistični element manjšamo. To je priročno v primerih, ko se pri vpisovanju statistike zmotimo in hočemo narediti popravek.

Pošiljanje sporočil strežniku

Sporočila, ki se pošiljajo iz mobilnih naprav na strežnik, vsebujejo trenutni rezultat na tekmi in imeni obeh moštev. Primer sporočila:

```
Ime prve ekipe 88 : 77 Ime druge ekipe
```

Pri pisanju statistike, se sporočilo ne pošlje ob vsakem dogodku, ampak le pri spremembi rezultata. Glavni razlog za to je, da večino ljudi, ki spremljajo neko tekmo, naj bo to v živo, prek interneta ali kako drugače, najbolj zanima rezultat tekme oziroma kako kaže njihovi ekipi.

Za pošiljanje sporočil v mobilni aplikaciji skrbi metoda sendDataToPhpServer, ki smo ji kot argument podali niz znakov, ki predstavljajo sporočilo. S strežnikom, ki smo ga postavili na naslovu http://basketball-statistics.site11.com, smo najprej vzpostavili povezavo, nato pa smo mu poslali podatke.

3.1.3 Razvoj aktivnosti za prejemanje sporočil od GCM strežnika

Sporočilo, ki je bilo poslano iz mobilne naprave na naš strežnik, se nato s strežnika pošlje naprej na GCM strežnik. Šele od tukaj se sporočilo pošlje na mobilne naprave (mobilne naprave prejmejo sporočilo). Da lahko mobilna naprava preko nameščene aplikacije prejme in obdela sporočilo od GCM strežnika, mora biti izpolnjenih več stvari:

- GCM strežnik mora vedeti, katerim napravam naj pošlje sporočilo
- nameščena aplikacija mora biti pravilno nastavljena, da lahko prejme sporočilo
- nameščena aplikacija mora vsebovati razred GCMIntentService.

V nadaljevanju si bomo pogledali, kaj točno je potrebno narediti, da zadostimo vsaki od omenjenih stvari.

GCM strežnik ne ve, katerim napravam naj pošlje sporočilo, zato mu mora to sporočiti naš strežnik. Iz tega sledi, da mora naš strežnik hraniti registracijske številke naprav, katerim naj se sporočilo pošlje. Mobilna naprava registracijsko številko dobi tako, da se registrira za GCM storitev. Dobljeno številko nato preko aplikacije pošlje našemu strežniku, ki številko shrani. Registracijsko številko aplikacija dobi na naslednji način:

```
String registrationId = GCMRegistrar.getRegistrationId(Games.this);
```

GCMRegistrar je razred, ki ima več metod, med njimi pa je tudi metoda getRegistrationId. Metoda kot odgovor vrne registracijsko številko.

Prejemanje sporočil od GCM strežnika aplikaciji omogočimo tako, da v nastavitvah (v projektu je to datoteka AndroidManifest.xml) dodamo oziroma določimo naslednje stvari:

- minimalno SDK različico, ki jo aplikacija podpira (to je potrebno zato, ker GCM storitev deluje le na verzijah operacijskega sistema, ki uporabljajo SDK različico 8 oziroma višjo):
  - `<uses-sdk android:minSdkVersion="8" android: targetSdkVersion="16"/>`
- aplikaciji dodamo posebna dovoljenja, da bo dobljena sporočila lahko prejela le ta aplikacija:
  - `<permission android:name="my_app_package.permission. C2D_MESSAGE" android:protectionLevel="signature" />`
  - `<uses-permission android:name="my_app_package.permission.C2D_MESSAGE" />`
- aplikaciji dodamo še naslednja dovoljenja:
  - dovoljenje za prejemanje GCM sporočil

    `<uses-permission android:name="com.google.android.c2dm.permission.RECEIVE" />`
  - dovoljenje za vzpostavitev internetne povezave in s tem povezave do Google storitev

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

  - dovoljenje za uporabo Google računa

    ```
    <uses-permission
    android:name="android.permission.GET_ACCOUNTS" />
    ```

  - dovoljenje oziroma nastavitev, ki procesorju ne dovoli da zaspi pri prejemanju sporočila

    ```
    <uses-permission
    android:name="android.permission.WAKE_LOCK" />
    ```

- Definiramo, katera storitev oziroma kateri razred skrbi za prejemanje in obdelavo sporočil (v temu primeru je to razred GCMIntentService):

  - `<service android:name=".GCMIntentService" />`

Kot rečeno, razred GCMIntentService skrbi za prejemanje in obdelavo sporočil, poleg tega pa določa še druge stvari. Metode, ki jih vsebuje razred, so:

- `onRegistered(Context context, String regId)`

  Metoda se kliče pri postopku registracije mobilne naprave za GCM storitev.

- `onUnregistered(Context context, String regId)`

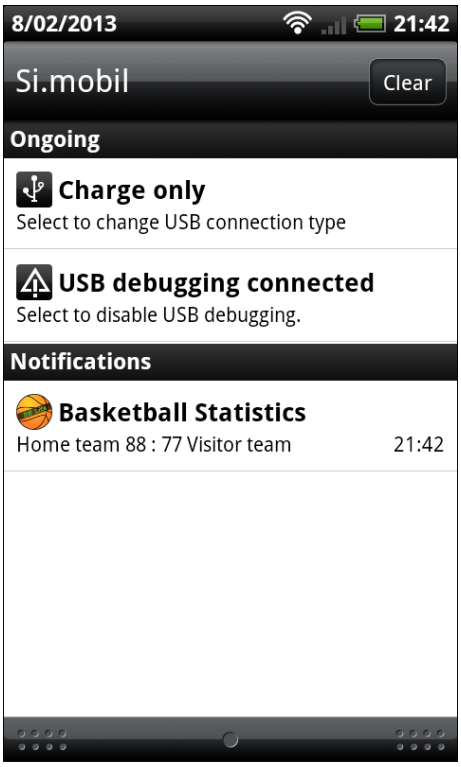
  Metoda se kliče pri odjavi mobilne naprave od GCM storitve.

- `onMessage(Context context, Intent intent)`

  Metoda se kliče, ko naš strežnik pošlje sporočilo GCM strežniku, ta pa ga dostavi do mobilne naprave.

- `onError(Context context, String errorId)`

  Metoda se kliče, ko se naprava hoče registrirati ali odjaviti od GCM storitve, vendar pri temu pride do napake.



Slika 3.4: Primer prejetega sporočila o rezultatu tekme.

- `onRecoverableError(Context context, String errorId)`

  Metoda se kliče, ko se naprava hoče registrirati ali odjaviti od GCM storitve, vendar GCM strežniki niso na voljo. V temu primeru je to napaka, ki se jo da odpraviti tako, da z registracijo ali odjavo poskušamo še enkrat oziroma večkrat, dokler nam ne uspe.

V našem primeru smo za boljšo preglednost razredu GCMIntentService dodali še metodo, ki skrbi za prikaz dobljenih sporočil. Metoda se kliče iz zgoraj omenjene metode onMessage, sporočila pa se na mobilni napravi prikažejo v vrstici stanja. Primer prejetega sporočila lahko vidimo na sliki 3.4.

Poglavje 4

Strežniški del

Strežniški del sistema je napisan v jeziku PHP in je postavljen na naslovu http://basketball-statistics.site11.com. Postavitev strežnika smo izvedli preko strani http://www.000webhost.com. Strežnik ima kot del sistema dve nalogi:

- shranjevanje registracijskih številk naprav, ki jim je sporočilo potrebno poslati
- prepošiljanje od mobilnih naprav dobljenih sporočil.

V nadaljevanju si bomo obe nalogi pogledali bolj podrobno.

Shranjevanje registracijskih številk naprav, ki jim je sporočilo potrebno poslati

Kot smo že omenili, mora strežnik hraniti registracijske številke naprav, ki jim bo sporočilo poslano. Strežnik namreč iz neke mobilne naprave dobi samo sporočilo, sam pa mora vedeti, katerim napravam bo sporočilo poslano naprej. V resnici strežnik sam od sebe ne ve, katere naprave naj prejmejo sporočilo, zato se morajo naprave predhodno prijaviti za prejemanje sporočil. V ta namen strežniku pošljejo zahtevek, kjer mu podajo naslednja podatka:

- registracijski enolični identifikator
- enolični identifikator naprave

Registracijski enolični identifikator naprava prejme pri registraciji na GCM storitev in je potreben, da se sporočilo pošlje na pravo napravo.

Enolični identifikator naprav (v našem primeru številka IMEI) je potreben za lastne potrebe strežnika. Pomemben je v primeru, če bi naprava neprestano pošiljala zahtevke na strežnik, da bi rada prejemala sporočila. V temu primeru se napravo samo enkrat vpiše v podatkovno bazo za prejemanje sporočil, v vseh ostalih primerih pa se zahtevek ignorira in le sporoči, da je naprava že registrirana.

Podatkovna baza za shranjevanje registracijskih številk je povsem enostavna. Vsebuje le eno tabelo za shranjevanje zgoraj omenjenih podatkov.

Prepošiljanje od mobilnih naprav dobljenih sporočil

Strežnik dobljenih sporočil ne more poslati direktno na mobilne naprave, ki so prijavljene na prejemanje sporočil. Dobljena sporočila mora skupaj s seznamom naprav, ki naj sporočilo prejmejo, poslati na GCM strežnik. Šele od tukaj se sporočila dejansko pošljejo na mobilne naprave.

Po prejemu zahtevka strežnik preveri ali gre za zahtevek tipa POST in prebere podatke (tj. trenutni rezultat tekme). Nato pridobi seznam naprav, ki naj podatke prejmejo. Seznam dobimo iz podatkovne baze na strežniku, iz katere preberemo registracijske številke naprav.

Iz podatkov, ki jih imamo (prejemniki, sporočilo), sestavimo sporočilo, organizirano v obliko JSON. Poleg omenjenih podatkov sporočilu dodamo še dodatno polje:

- collapse_key - polje je pomembno v naslednjem primeru: Neka naprava je prijavljena na prejemanje sporočil, vendar izgubi povezavo z internetom. Če polje collapse_key ni nastavljeno, bo naprava, ko bo internetna

povezava ponovno vzpostavljena, dobila vsa sporočila, ki so ji bila poslana med tem časom. V nasprotnem primeru, če polje collapse_key je nastavljeno, bo naprava dobila le zadnje sporočilo. V našem primeru je to povsem smiselno, saj uporabnika zanima zadnji rezultat na tekmi, in ne še vsi vmes.

V kodi JSON sporočilo sestavimo takole:

```
$fields = json_encode(array(
     'registration_ids'  => $this->devices,
     'data'              => $data,
     'collapse_key'=> "collapseKey"
));
```

Ko imamo sporočilo sestavljeno, moramo nastaviti še pravilne vrednosti v glavi sporočila. Povedati moramo, da so podatki v obliki JSON in nastaviti API ključ strežnika. API ključ je shranjen na strežniku, potreben pa je, da GCM strežnik spozna naš strežnik. GCM strežnik namreč zahtevkov od strežnikov, ki nimajo pravega API ključa, ne sprejema. API ključ se pridobi preko Googlove API konzole. Glavo sporočila v kodi nastavimo takole:

```
$headers = array(
     'Authorization: key=' . $this->serverApiKey,
     'Content-Type: application/json'
);
```

Ko je to opravljeno, je sporočilo potrebno samo še poslati GCM strežniku. Tukaj se vloga našega strežnika konča. Ostalo delo, ki se tiče dostavljanja sporočil na mobilne naprave, mora opraviti GCM strežnik.

Poglavje 5

Zaključek

V času pisanja diplomske naloge smo razvili aplikacijo za pisanje statistike košarkarskih tekem in spremljanje rezultatov več tekem v živo, če to želimo. Aplikacija je napisana za uporabo na mobilnem operacijskem sistemu Android in jo je možno naložiti na naprave z Androidom različice 2.2 oziroma novejše. Razlog za to je uporaba GCM storitve, ki smo jo uporabili za pošiljanje sporočil (tj. rezultatov tekem) vsem uporabnikom, ki želijo spremljati rezultate tekem v živo.

Med razvijanjem sistema smo veliko pozornosti posvetili tako izgledu aplikacije kot tudi prijetni uporabniški izkušnji. Aplikacijo smo zasnovali na način, da se že ob prvem stiku z njo počutimo domače in točno vemo, kaj katera stvar naredi. Menim, da nam je to uspelo uspešno narediti. Sicer nam je največ problemov povzročala implementacija GCM storitve. Potrebno je bilo pravilno nastaviti vse stvari v povezavi z aplikacijo, kar ni bilo enostavno. Dokumentacija GCM storitve sicer vsebuje navodila, kaj je potrebno v projektu dodati ali spremeniti, vendar je vse napisano po delih. Pogrešali smo primer, s katerim bi bilo vse skupaj zajeto v delujočo celoto, ki bi služila le kot osnova za nadaljnji razvoj.

Trenutno lahko uporabniki aplikacije spremljajo rezultate tekem v živo tako, da dobijo obvestilo o rezultatu v statusno vrstico mobilne naprave. To

bi bilo možno izboljšati na način, da bi razvili novo aktivnost v aplikaciji, ki bi služila le prikazovanju rezultatov vseh tekem, saj bi tako veliko pridobili na preglednosti. Z razvojem take aktivnosti bi lahko dodali tudi filtre, ki bi prikazali le tekme, ki nas zanimajo.

Slike

2.1 Razširjenost posameznih različic med Android napravami. . . . 7
2.2 Arhitektura operacijskega sistema Android. 8
2.3 Življenjski cikel aktivnosti. 13
2.4 Facebook - primer PUSH tehnologije. 18
2.5 GCM storitev - osnovno delovanje. 20
2.6 PHP - osnovno delovanje. 32
2.7 Android emulator na operacijskem sistemu Windows. 33
2.8 Struktura projekta v razvojnem orodju Eclipse pri razvoju Android aplikacije. 34

3.1 Vstopna aktivnost aplikacije. 37
3.2 Model podatkovne baze za aplikacijo. 38
3.3 Aktivnost za pisanje statistike. 42
3.4 Primer prejetega sporočila o rezultatu tekme. 47

Tabele

2.1 Razširjenost posameznih različic operacijskega sistema Android. 6
2.2 Ključni pojmi in koncepti vključeni v GCM. 22
2.3 Možni statusi HTTP odziva pri pošiljanju sporočila. 29
2.4 Polja prisotna v telesu pozitivnega odziva po JSON zahtevi. . . 30

Literatura

[1] (2013) Android Developers. Dostopno na:
http://developer.android.com/index.html

[2] (2013) Android (operating system). Dostopno na:
http://en.wikipedia.org/wiki/Android_(operating_system)

[3] (2013) FRI - Android Wiki. Dostopno na:
http://android.fri.uni-lj.si/index.php/Glavna_stran

[4] (2013) Google Play Has 700,000 Apps, Tying Apple's App Store. Dostopno na:
http://mashable.com/2012/11/01/google-apps-tie-apple/

[5] (2013) Pull technology. Dostopno na:
http://en.wikipedia.org/wiki/Pull_technology

[6] (2013) Push technology. Dostopno na:
http://en.wikipedia.org/wiki/Push_technology

[7] (2013) Google Cloud Messaging for Android. Dostopno na:
http://developer.android.com/google/gcm/index.html

[8] (2013) PHP. Dostopno na:
http://en.wikipedia.org/wiki/PHP

[9] (2013) MySQL. Dostopno na:
http://sl.wikipedia.org/wiki/MySQL