Bluetooth, BLE, NFC i Nearby API

prof. dr Bratislav Predić
dipl. inž. Nevena Tufegdžić

Razvoj mobilnih aplikacija i servisa
Računarstvo i informatika

2024.

Bluetooth

prof. dr Bratislav Predić
dipl. inž. Nevena Tufegdžić

BT, BLE, NFC i Nearby API
Razvoj mobilnih aplikacija i servisa

Bluetooth

- Android platforma ima podršku za Bluetooth network stack, koji omogućava bežični prenos podataka
- Pristup Bluetooth-u je omogućen pomoću Bluetooth API-ja
- Pomoću Bluetooth API-ja moguće je:
  - Skenirati okolinu za Bluetooth uređaje
  - Preuzeti listu trenutno povezanih Bluetooth uređaja
  - Povezati se sa drugim uređajima
  - Prenositi podatke na i sa drugih uređaja

Bluetooth

- Proces uparivanja dva uređaja se sastoji iz više koraka:
  - Jedan uređaj (A) pokreće proces koji omogućava drugim uređajima da ga otkriju i postaje dostupan za dolazne zahteve za povezivanje
  - Drugi uređaj (B) pronalazi uređaj A koristeći proces otkrivanja (service discovery), i šalje zahtev za uparivanje
  - Uređaj A prihvata zahtev za uparivanje nakon kog uređaji međusobno razmene sigurnosne ključeve koji se keširaju radi kasnijeg korišćenja

Bluetooth

- Nakon što dva uređaja prođu proces uparivanja, spremni su za proces razmene podataka
- Po završetku procesa razmene podataka, uređaj koji je inicirao konekciju (B) oslobađa kanal
- Uređaji ostaju upareni nakon završetka celog procesa i mogu se ponovo povezati ukoliko u budućnosti postoji potreba da se ponovo prenesu podaci
- Ukoliko jedan uređaj ukloni vezu (u podešavanjima) uređaji će morati ponovo da prođu proces uparivanja

Bluetooth

- Aplikacija započinje proces korišćenja Bluetooth-a preko BluetoothAdapter-a
- Preko ove klase, aplikacija proverava da li je Bluetooth dostupan na uređaju

Uspostavljanje veze

- Ako je Bluetooth dostupan, postoje tri koraka za uspostavljanje veze:
  - Pronalazak Bluetooth uređaja u blizini – već uparenih ili novih
  - Povezivanje sa Bluetooth uređajem
  - Prenos podataka sa povezanim uređajem

Bluetooth API

- Bluetooth API ima sledeće glavne komponente:
  - BluetoothAdapter
  - BluetoothDevice
  - BluetoothSocket
  - BluetoothServerSocket
  - BluetoothClass
  - BluetoothProfile

BluetoothAdapter

- BluetoothAdapter predstavlja lokalni Bluetooth adapter na uređaju
- Ovo je početna tačka za sve Bluetooth interakcije
- Koristi se radi pretrage Bluetooth uređaja, pribavljanje liste uparenih uređaja i kreiranje socket-a za osluškivanje komunikacija od drugih uređaja

BluetoothDevice

- BluetoothDevice predstavlja udaljen Bluetooth uređaj
- Koristi se radi slanja zahteva za konekciju sa udaljenim uređajem preko BluetoothSocket-a ili upita informacija o uređaju (ime, adresa, klasa, stanje veze)

BluetoothSocket

- BluetoothSocket je interfejs za soket preko kog se vrši komunikacija
- To je tačka veze koja omogućava aplikaciji da razmenjuje podatke sa drugim Bluetooth uređajem

BluetoothServerSocket

- BluetoothServerSocket predstavlja otvoreni server socket koji osluškuje dolazne zahteve za konekciju
- Kako bi se dva uređaja povezala, jedan uređaj mora otvoriti server socket sa ovom klasom
- Kada udaljeni uređaj uspostavi vezu, uređaj prihvata konekciju i vraća povezani BluetoothSocket

BluetoothClass i BluetoothProfile

- BluetoothClass – opisuje karakteristike i mogućnosti Bluetooth uređaja.
  - Read-only skup svojstava koji definišu klase i servise uređaja
- BluetoothProfile – interfejs koji predstavlja Bluetooth profil (specifikacija bežičnog interfejsa za Bluetooth baziranu komunikaciju između uređaja)

Podešavanje Bluetooth-a

- Pre nego što aplikacija krene sa Bluetooth komunikacijom, potrebno je izvšiti proveru da li je Bluetooth podržan na uređaju
- Prvi korak predstavlja dodavanje Bluetooth permisija u manifest fajlu
- Nakon dodavanja permisija, podešavanje Bluetooth-a se obavlja u dva koraka koristeći BluetoothAdapter:
  1. Pribavljanje BluetoothAdapter-a
  2. Uključivanje Bluetooth-a

Podešavanje Bluetooth-a

```
// KORAK 1
val bluetoothManager: BluetoothManager =
        getSystemService(BluetoothManager::class.java)
val bluetoothAdapter: BluetoothAdapter? = bluetoothManager.getAdapter()
if (bluetoothAdapter == null) {
  // Uređaj ne podržava Bluetooth
}
// KORAK 2
if (bluetoothAdapter?.isEnabled == false) {
    val enableBtIntent = Intent(BluetoothAdapter.ACTION_REQUEST_ENABLE)
    startActivityForResult(enableBtIntent, REQUEST_ENABLE_BT)
}
```

Pronalazak Bluetooth uređaja

- Proces pronalaska novih uređaja predstavlja proceduru gde uređaj skenira okolinu za Bluetooth uređaje koji su upaljeni
- Uređaj započinje proceduru slanjem zahteva svim Bluetooth uređajima u okolini
- Na zahtev odgovaraju samo oni uređaji koji su u otkrivenom (discoverable) modu
- Odgovor na zahtev predstavlja informacija o uređaju (ime uređaja, klasa, MAC adresa)

Pronalazak Bluetooth uređaja

- Ovaj proces se preko Bluetooth API-ja obavlja u više koraka:
  - Pretraga već uparenih uređaja – treba proveriti da li je traženi uređaj već uparen sa trenutnim pre pokretanja procesa pretrage novih uređaja
  - Pretraga novih uređaja – proces pokretanja pretrage se obavlja pozivom startDiscovery() metode Bluetooth adapter klase. Registrovanjem ACTION_FOUND intent-a pomoću BroadcastReceiver-a moguće je dobiti informacije o uređajima koji su odgovorili na poslate zahteve
  - Omogućavanje vidljivosti (discoverability) lokalnog uređaja drugim uređajima – pozivom startActivityForResult sa ACTION_REQUEST_DISCOVERABLE intentom

Pronalazak Bluetooth uređaja

```
// Pribavljanje već uparenih uređaja
val pairedDevices: Set<BluetoothDevice>? =
bluetoothAdapter?.bondedDevices
pairedDevices?.forEach { device ->
    val deviceName = device.name
    val deviceHardwareAddress = device.address // MAC addr
}
```

Pronalazak Bluetooth uređaja

```
override fun onCreate(savedInstanceState: Bundle?) {
    ...

    // Registrovanje na događaj kada se uređaj pronađe
    val filter = IntentFilter(BluetoothDevice.ACTION_FOUND)
    registerReceiver(receiver, filter)

    // Pokretanje procesa pretrage novih uređaja
    bluetoothAdapter.startDiscovery()
}
```

Pronalazak Bluetooth uređaja

```
// BroadcastReceiver za ACTION_FOUND
private val receiver = object : BroadcastReceiver() {
    override fun onReceive(context: Context, intent: Intent) {
        val action: String = intent.action
        when(action) {
            BluetoothDevice.ACTION_FOUND -> {
                val device: BluetoothDevice =
                        intent.getParcelableExtra(
                                BluetoothDevice.EXTRA_DEVICE
                         )
                val deviceName = device.name
                val deviceHardwareAddress = device.address // MAC addr
            }
        }
    }
}
```

Povezivanje Bluetooth uređaja

- Kako bi se omogućila konekcija između dva uređaja potrebno je implementirati i klijent i server mehanizme
- Jedan uređaj otvara server socket, a drugi inicira konekciju na osnovu MAC adrese servera
- Server pribavlja informacije o socket-u kada je konekcija prihvaćena, dok klijent pruža informacije o socket-u kada otvori RFCOMM kanal prema serveru
- Klijent i server su povezani međusobno kada imaju povezan BluetoothSocket na istom RFCOMM kanalu

Povezivanje Bluetooth uređaja

- Konekcija se može implementirati na dva načina:
  - Oba uređaja mogu da se podese kao server i da osluškuju za konekciju - u ovom slučaju bilo koji uređaj može da započne konekciju i da postane klijent
  - Alterantivno, jedan uređaj može biti server, dok će drugi otvoriti konekciju

Povezivanje Bluetooth uređaja

- Kada uređaji koji nisu upareni započnu konekciju počinje proces uparivanja
- Android Framework automatski prikazuje dijaloge koji omogućavaju uparivanje uređaja
- Klijent i server je potrebno implementirati u nitima jer sadrže blokirajuće pozive

Prenos podataka

- Nakon što je uspostavljena konekcija između dva uređaja, moguće je izvršiti prenos podataka
- Proces prenosa podataka se sastoji iz više koraka:
  - Pribavljanje InputStream i OutputStream objekta koji su zaduženi za transmisiju podataka preko socket-a pomoću getInputStream() i getOutputStream() metoda socket-a
  - Čitanje i pisanje podataka u stream-ove koristeći read(byte[]) metode InputStream-a i write(byte[])) metode OutputStream-a
  - read() i write() metode su blokirajuće i zbog toga je potrebno smestiti funkcije koje obrađuju transmisiju podataka u posebne niti

Bluetooth Low Energy

- Bluetooth Low Energy (BLE) je tehnologija koja omogućava prenos male količine podataka uz minimalnu potrošnju baterije
- Koristi se za komunikaciju sa uređajima koji imaju striktne zahteve za strujom, kao što su monitori pulsa i fitnes uređaji
- Za implementaciju BLE komunikacije koristi se Bluetooth LE API

Korišćenje BLE

- Uređaji koji žele da komuniciraju preko BLE moraju prvo formirati konekciju u vidu komunikacionog kanala
- Pre implementacije BLE u aplikaciji, potrebno je deklarisati Bluetooth permisije u manifest fajlu
- Nakon deklarisanja permisija, potrebno je pristupiti BluetoothAdapter-u kako bi se proverilo da li je Bluetooth dostupan na uređaju

Korišćenje BLE

- Ukoliko je Bluetooth dostupan na uređaju, uređaj skenira okolinu i pronalazi dostupne BLE uređaje
- Kada se pronađe BLE uređaj na koji je potrebno izvršiti konekciju, atributi tog uređaja čitaju se pomoću konektovanja na njegov GATT server
- Kada se uspostavi konekcija, moguć je prenos podataka u skladu sa karakteristikama i mogućnostima BLE uređaja

GATT server

- Generic ATTribute profile je specifikacija profila uređaja koja prikazuje način slanja i primanja podataka kroz BLE vezu
- Podaci koji se šalju preko BLE nazivaju se atributima
- Svi profili BLE aplikacija bazirani su na GATT-u
- Konekcija na BLE uređaj vrši se konektovanjem na njegov GATT server

Near Field Communication

prof. dr Bratislav Predić
dipl. inž. Nevena Tufegdžić

BT, BLE, NFC i Nearby API
Razvoj mobilnih aplikacija i servisa

NFC

- NFC predstavlja bežičnu tehnologiju za veoma blisku komunikaciju
- Maksimalna razdaljina uređaja koja je potrebna za komunikaciju je tipično 4cm ili manje
- NFC služi za prenos manje količine podataka
- Prenos se vrši između NFC taga i Android uređaja ili između dva Android uređaja

Omogućavanje NFC-a

- Kako bi NFC funkcionalnosti bile dostupne aplikaciji, potrebno je u AndroidManifest.xml fajlu dodati sledeće:

```
<uses-permission android:name="android.permission.NFC" />
<uses-sdk android:minSdkVersion="10"/>
<uses-feature android:name="android.hardware.nfc"
        android:required="true" />
```

NFC tag

- NFC tag je način kodiranja NFC tehnologije tako da omoguće bežičnu komunikaciju sa nekim uređajem
- NFC tagovi mogu da komuniciraju sa mobilnim telefonima, laptopovima, tablet računarima i drugim elektronskim uređajima koji implementiraju NFC tehnologiju
- NFC tag nije isto što i RFID (Radio-Frequency Identification)

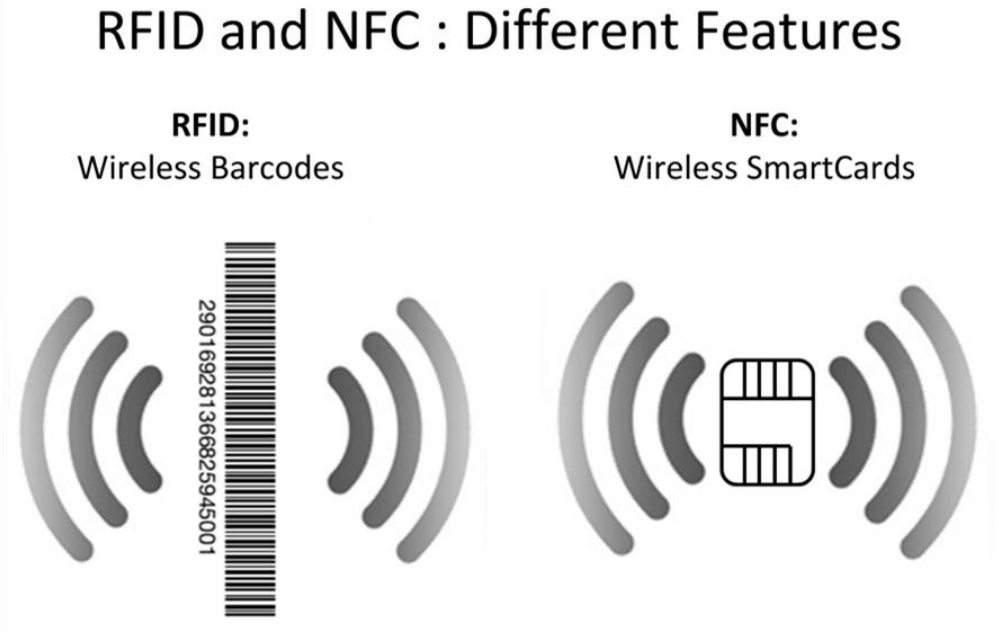
NFC vs RFID

- RFID je prethodnik NFC-a
- Kada nešto ima RFID implementirano, podržava samo jednosmernu komunikaciju – uređaj sa RFID-jem može samo da komunicira sa RFID čitačem
- RFID služi za implementaciju sistema za sprečavanje krađa, kao i sistema za praćenje stvari

NFC vs RFID

- NFC je delom zasnovan na RFID-ju, a delom na Bluetooth tehnologiji
- Omogućava dvosmernu komunikaciju između čipova
- Za razliku od RFID-ja, NFC zahteva da su čipovi fizički blizu
- Za razliku od BLE, NFC ne zahteva ručno otkrivanje i sinhronizaciju uređaja koji treba da komuniciraju

NFC vs RFID


RFID and NFC : Different Features
RFID:
Wireless Barcodes
NFC:
Wireless SmartCards
2901692813668259450 01

NFC tag

- Kompleksnost NFC tagova varira – mogu da omogućavaju jednostavne read-write opcije, sa one-time-programmable delovima čipa tako da kartica bude read-only
- Kompleksnije implementacije tagova nude i matematičke funkcije sa kriptovanim delovima kako bi se osigurala autentifikacija
- Podaci u NFC tagu poštuju NDEF (NFC Data Exchange Format) standard

Android i NFC

- Android uređaji sa implementiranim NFC-om podržavaju sledeće načine funkcionisanja:
  - **Reader/writer mode** – NFC uređaj može da čita i upisuje pasivne NFC tagove
  - **P2P mode** – NFC uređaj može da razmenjuje podatke sa drugim NFC uređajem (npr. Android Beam)
  - **Card emulation mode** – NFC uređaj se ponaša kao NFC kartica kojoj može da pristupi neki NFC reader

Android i NFC

- Svaka NFC operacija u Androidu mora da poštuje NDEF standard i NFC poruke su zapravo NDEF poruke
- Pri implementaciji NFC-a u aplikacijama postoje dva glavna use case-a:
  - Čitanje NDEF podataka sa NFC taga
  - Beam-ovanje NDEF poruka nekom NFC uređaju (Android Beam)

Čitanje NFC taga

- Čitanje sa NFC taga vrši se uz pomoć tag dispatch sistema
- Tag dispatch sistem skenira i vrši analizu otkrivenih NFC tagova, i na osnovu toga kategorizuje podatke u njima
- Nakon kategorizacije podataka, tag dispatch sistem startuje aplikaciju koja je zainteresovana za te podatke
- Aplikacija koristi Intent Filter kako bi rekla tag dispatch sistemu da je zainteresovana za obradu određene kategorije podataka

Čitanje NFC taga

- Obrada NFC taga vrši se kroz tri koraka:
  1. Parsiranje NFC taga – određivanje MIME tipa ili URI-ja koji identifikuje podatke u tagu
  2. Kreiranje intenta – enkapsulacija MIME tipa ili URI-ja u intent
  3. Startovanje Activity-ja koji odgovara datom intentu

Parsiranje NFC taga

- NFC tag sadrži podatke pisane u NDEF formatu
- NDEF podaci su enkapsulirani u okviru poruke (NdefMessage) koja sadrži jedan ili više record-a (NdefRecord)
- Svaki record mora da bude formatiran u skladu sa specifikacijom tipa sloga

NdefRecord

- Tipičan NdefRecord sadrži sledeća polja:
  - TNF (Type Name Format) – opisuje način interpretacije variable length type polja
  - Variable length type – opisuje tip record-a
  - Variable length ID – jedinstveni identifikator record-a (opciono)
  - Variable length payload – podaci koji se prenose

Parsiranje NFC taga

- Kada tag dispatch sistem otkrije tag pisan po NDEF standardu, proverava sadržaj prvog record-a u NdefMessage-u
- Na osnovu prvog NdefRecord-a, tag dispatch sistem definiše način obrade čitavog taga
- Koriste se TNF i variable length type polja kako bi se tag mapirao u MIME tip ili URI

Parsiranje NFC taga

- Najčešći primeri parsiranja:
  - TNF polje ima vrednost TNF_ABSOLUTE_URI – vrši se mapiranje u URI na osnovu type polja
  - TNF ima vrednost TNF_MIME_MEDIA – vrši se mapiranje u MIME tip na osnovu vrednosti type polja
- Kompletna lista mogućih TNF vrednosti može se naći na: https://developer.android.com/guide/topics/connectivity/nfc/nfc#ndef

Prosleđivanje NFC tagova aplikacijama

- Nakon što tag dispatch sistem identifikuje tip taga i parsira ga, kreira se Intent koji enkapsulira tag i informacije o njemu
- Intent se šalje aplikaciji koja je registrovana da obrađuje takav Intent pomoću njenog IntentFilter-a
- Ukoliko više aplikacija može da obradi Intent, otvara se Activity Chooser
  - Ovo je loše jer NFC treba da radi bez intervencije korisnika, tako da pri pisanju aplikacija treba voditi računa o ovome

Prosleđivanje NFC tagova aplikacijama

- Tag dispatch sistem definiše tri tipa Intent-a, poređanih od najvećeg ka najmanjem prioritetu:
  1. ACTION_NDEF_DISCOVERED
  2. ACTION_TECH_DISCOVERED
  3. ACTION_TAG_DISCOVERED

Prosleđivanje NFC tagova aplikacijama

- **ACTION_NDEF_DISCOVERED** je Intent najvišeg prioriteta i Android sistem pokušava da pokrene Activity sa ovim Intentom pre svih ostalih
- Ovaj Intent se pravi kada se otkrije tag pisan u NDEF formatu i čiji je tip prepoznat

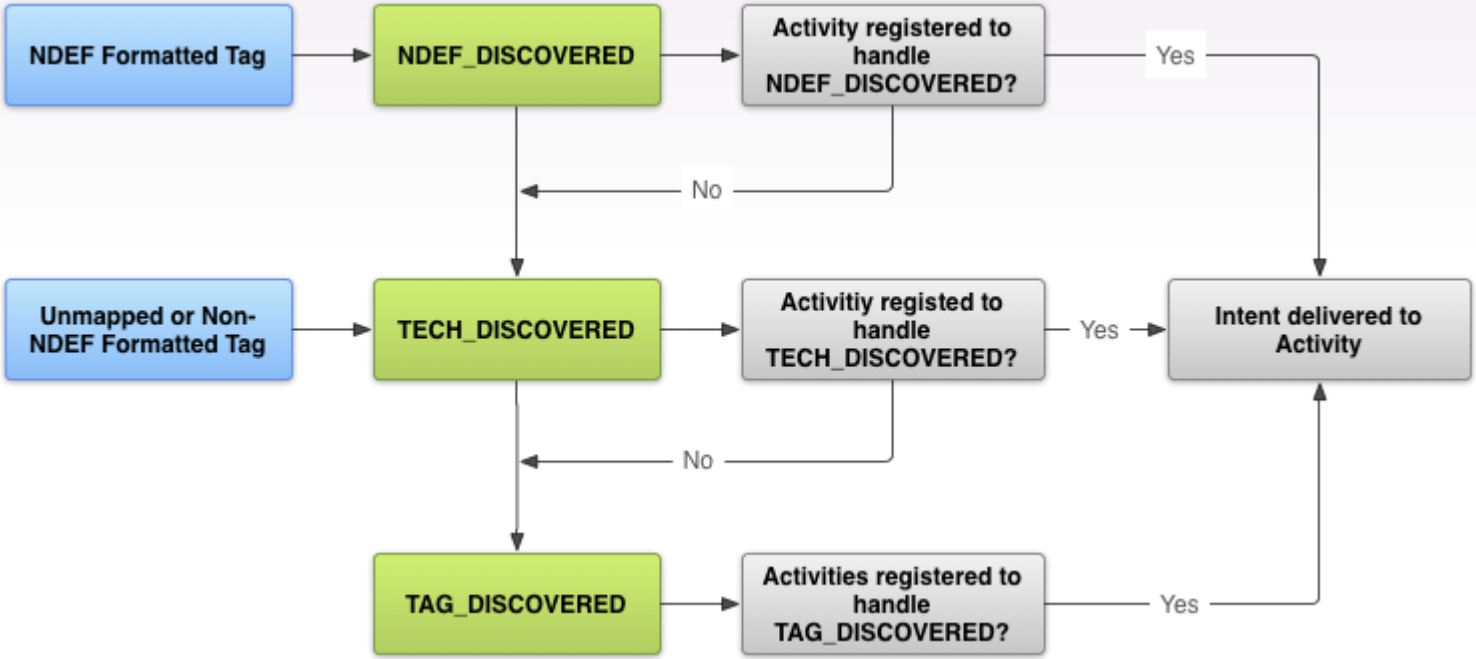
Prosleđivanje NFC tagova aplikacijama

- **ACTION_TECH_DISCOVERED** se koristi onda kada ne postoji Activity koji je registrovan da obradi Intent sa ACTION_NDEF_DISCOVERED
- Ovaj Intent se pravi bez prethodnog kreiranja ACTION_NDEF_DISCOVERED kada se otkrije tag koji nije pisan u NDEF formatu ali čija je tehnologija prepoznata
- Druga varijanta kada se ovaj Intent odmah kreira je kada je tag pisan u NDEF formatu ali tip nije prepoznat

Prosleđivanje NFC tagova aplikacijama

- **ACTION_TAG_DISCOVERED** se koristi onda kada ne postoji Activity koji je registrovan da obradi Intent sa ACTION_NDEF_DISCOVERED ili sa ACTION_TECH_DISCOVERED

Prosleđivanje NFC tagova aplikacijama



Deklaracija IntentFilter-a

- **ACTION_NDEF_DISCOVERED** – MIME:

```
<intent-filter>
    <action android:name="android.nfc.action.NDEF_DISCOVERED"/>
    <category android:name="android.intent.category.DEFAULT"/>
    <data android:mimeType="text/plain" />
</intent-filter>
```

- Uz deklaraciju akcije potrebno je dodati i data format kako bi Intent obrađivao tačan MIME tip (u ovom slučaju to je text/plain)

Deklaracija IntentFilter-a

- **ACTION_NDEF_DISCOVERED** – URI:

```
<intent-filter>
    <action android:name="android.nfc.action.NDEF_DISCOVERED"/>
    <category android:name="android.intent.category.DEFAULT"/>
   <data android:scheme="https"
              android:host="developer.android.com"
              android:pathPrefix="/index.html" />
</intent-filter>
```

Preuzimanje informacija iz Intent-a

- Kada Activity dobije NFC Intent, može da preuzme informacije o skeniranom NFC tagu
- Mogući Extras:
  - EXTRA_TAG – obavezan je i sadrži Tag objekat koji je skeniran
  - EXTRA_NDEF_MESSAGES – niz NDEF poruka koje su izvedene iz taga
  - EXTRA_ID – opcioni ID taga

Beam-ovanje NDEF poruka

- Beam-ovanje je drugi način NFC komunikacije koji podrazumeva dva Android uređaja
- Aplikacija koja želi da beam-uje podatke se mora izvršavati u foreground-u, a uređaj koji prima beam ne sme da bude zaključan
- Kada se uređaj koji beamuje dovoljno približi drugom uređaju, na njemu se prikazuje poruka „Touch to Beam“

Beam-ovanje NDEF poruka

- Za omogućavanje beam-ovanja, potrebno je pozvati jednu od sledeće dve metode:
  - sendNdefPushMessage() – prihvata NdefMessage parametar i šalje je kada se uređaj nađe u blizini drugog NFC uređaja
  - setNdefPushMessageCallback() – prihvata callback koji sadrži createNdefMessage() metod koji kreira NdefMessage tek kada je uređaj u blizini drugog NFC uređaja

Primer

- Kreiranje Activity-ja koji vrši beam-ovanje:

```
class Beam : Activity(), NfcAdapter.CreateNdefMessageCallback {

    private var nfcAdapter: NfcAdapter? = null
    private lateinit var textView: TextView
```

Primer

```
override fun onCreate(savedInstanceState: Bundle?) {
    super.onCreate(savedInstanceState)
    setContentView(R.layout.main)
    textView = findViewById(R.id.textView)
    // Pribavljanje NFC adaptera
    nfcAdapter = NfcAdapter.getDefaultAdapter(this)
    if (nfcAdapter == null) {
        Toast.makeText(this, "NFC is not available", Toast.LENGTH_LONG).show()
        finish()
        return
    }
    // Registracija callback-a
    nfcAdapter?.setNdefPushMessageCallback(this, this)
}
```

Primer

```
override fun createNdefMessage(event: NfcEvent): NdefMessage {
    val text = "Beam me up, Android!\n\n" +
            "Beam Time: " + System.currentTimeMillis()
    return NdefMessage(
        arrayOf(
            createMime(
                "application/vnd.com.example.android.beam",
                text.toByteArray()
            )
        )
    )
}
```

Primer

```
override fun onResume() {
    super.onResume()
    // Provera da li je aktivnost startovana zbog Android Beam-a
    if (NfcAdapter.ACTION_NDEF_DISCOVERED == intent.action) {
        processIntent(intent)
    }
}

override fun onNewIntent(intent: Intent) {
    // Nakon ovoga se poziva onResume da bi obradila intent
    setIntent(intent)
}
```

Primer

```
private fun processIntent(intent: Intent) {
    textView = findViewById(R.id.textView)
    // Samo jedna poruka se šalje u beam-u
    intent.getParcelableArrayExtra(NfcAdapter.EXTRA_NDEF_MESSAGES)?.also {
        rawMsgs ->
        (rawMsgs[0] as NdefMessage).apply {
            // record 0 sadrži MIME tip
            textView.text = String(records[0].payload)
        }
    }
}
```

Nearby API

prof. dr Bratislav Predić
dipl. inž. Nevena Tufegdžić

BT, BLE, NFC i Nearby API
Razvoj mobilnih aplikacija i servisa

Nearby API

- Nearby API je platforma koja omogućava olakšanu konektivnost i komunikaciju sa drugim uređajima preko Bluetooth-a, WiFi-ja i IP-ja
- Sastoji se od:
  - Nearby Connections API
  - Nearby Messages API
  - Fast Pair

Nearby Connections API

- Nearby Connections API omogućava aplikacijama da se lako povezuju i vrše prenos podataka bez učešća i potrebe za Internet konekcijom
- Koristi Bluetooth tehnologije i namenjen je da sakrije kompleksnosti implementacije
- Fokus je od pisanja koda za konekciju prebačen na interakciju sa drugim uređajima

Nearby Connections API

- Nearby Connections API je koristan za:
  - Otkrivanje uređaja u blizini bez potrebe za Internetom i uz izbegavanje cene round-trip mrežne komunikacije za svaku željenu oepraciju
  - Prenos proizvoljne količine podataka ili fajlova između bliskih uređaja
  - Ako se implementacija vrši isključivo na Android uređajima

Nearby Messages API

- Nearby Messages API omogućava prenos malih poruka pomoću publish-subscribe sistema koji se odvija preko cloud-a
- Uređaji koji koriste Nearby Messages API primaju poruke ukoliko su subscribed i u blizini uređaja koji vrši publish tih poruka
- Nearby uređaji se detektuju korišćenjem Bluetooth-a i ultrazvučne detekcije

Nearby Messages API

- Nearby Messages API je dobar za:
  - Otkrivanje uređaja u blizini dok je uređaj povezan na Internet
  - Jednosmerni prenos male količine podataka od uređaja u blizini
  - Skeniranje za BLE Beacons
  - Ukoliko se komunikacija vrši između Android i iOS uređaja

Fast Pair

- Fast Pair servis omogućava komunikaciju pomoću BLE tehnologije sa što manje korisničkih intervencija u toku uparivanja
- Koristi se za Bluetooth zvučnike, pametne narukvice, Bluetooth slušalice i sl.

Literatura

- Bluetooth overview | Android Developers
- Bluetooth Low Energy | Android Developers
- Near field communication overview | Android Developers
- https://www.nomtek.com/blog/what-are-nfc-tags
- Nearby | Google for Developers

Hvala na pažnji!