# Seminar „Android“
# Thema: Schnittstellen

von Christian Baumann

# Inhalt

1. Überblick Schnittstellen
   - Welche Schnittstellen gibt es?
   - Was ist das besondere an Schnittstellen bei Android?
2. Content Provider
   - Warum Content Provider
   - Beispiele für existierende Content Provider
   - Funktionsprinzip der Content Provider
   - Sicherheit
   - Vor- und Nachteile
   - Alternativen zum Content Provider
3. Netzwerk-Schnittstellen
   - Besonderheiten von Netzwerken bei mobilen Geräten
   - Welche Schnittstellen hält Android bereit?
   - Funktionsweise an einem Beispiel

# Inhalt

1. Überblick Schnittstellen
   - Welche Schnittstellen gibt es?
   - Was ist das besondere an Schnittstellen bei Android?
2. Content Provider
   - Warum Content Provider
   - Beispiele für existierende Content Provider
   - Funktionsprinzip der Content Provider
   - Sicherheit
   - Vor- und Nachteile
   - Alternativen zum Content Provider
3. Netzwerk-Schnittstellen
   - Besonderheiten von Netzwerken bei mobilen Geräten
   - Welche Schnittstellen hält Android bereit?
   - Funktionsweise an einem Beispiel

# Welche Schnittstellen gibt es?

- Grafik-Schnittstellen
- Eingabe-Schnittstellen
- Telefon-Schnittstellen
- Speicher-Schnittstellen
- Netzwerk-Schnittstellen

Android besitzt 394 Schnittstellen. Deshalb:

Beschränkung auf Speicher- und Netzwerk-Schnittstellen

# Besonderheiten der Schnittstellen

- Netzwerke
  - Android läuft auf mobilen Geräten
  - Große Vielzahl an möglichen Netzwerkverbindungen
  - Begrenzte Energieversorgung
  - Keine allgemeine Architektur
  - Nicht nachrüstbar
- Speicher
  - Begrenzter Speicher
  - Wechselnde Speichermedien
  - Ausführung von schädlicher Software (Spyware) möglich

# Inhalt

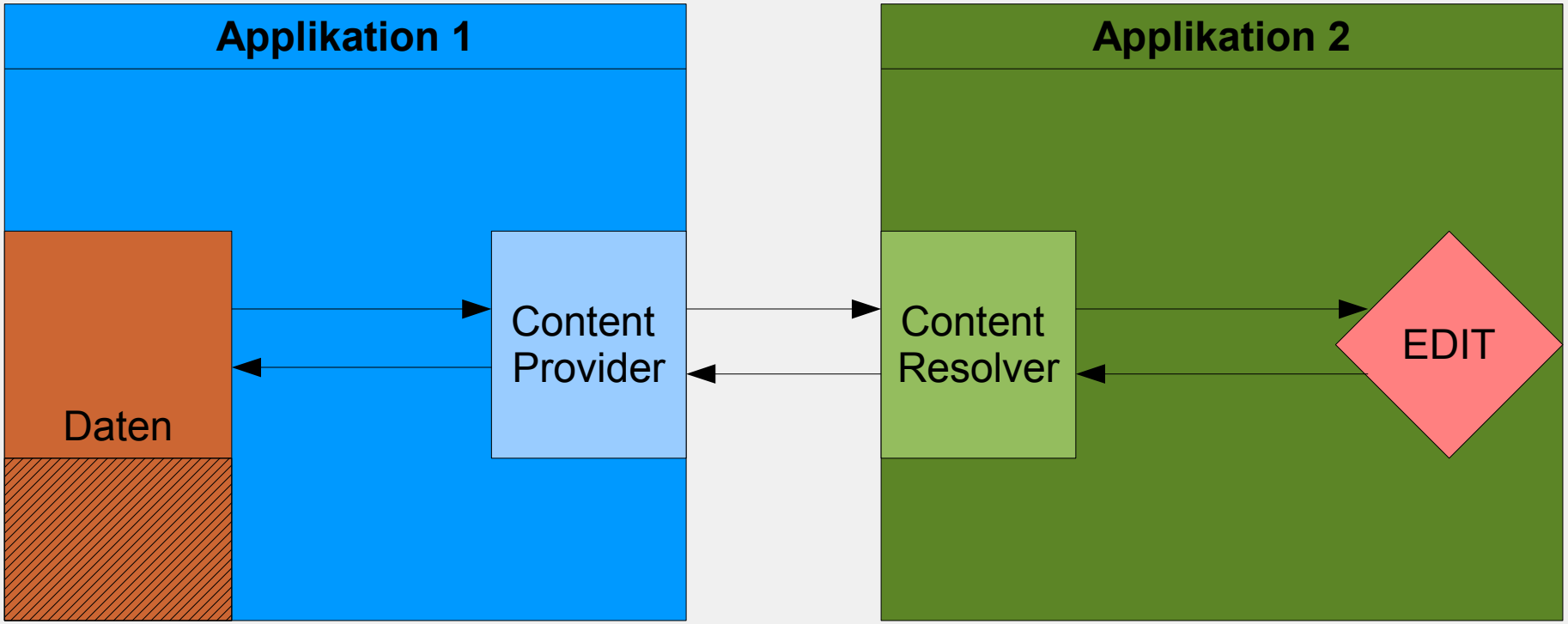
1. Überblick Schnittstellen
   - Welche Schnittstellen gibt es?
   - Was ist das besondere an Schnittstellen bei Android?
2. Content Provider
   - Warum Content Provider
   - Beispiele für existierende Content Provider
   - Funktionsprinzip der Content Provider
   - Sicherheit
   - Vor- und Nachteile
   - Alternativen zum Content Provider
3. Netzwerk-Schnittstellen
   - Besonderheiten von Netzwerken bei mobilen Geräten
   - Welche Schnittstellen hält Android bereit?
   - Funktionsweise an einem Beispiel

# Warum Content Provider?

- Alle Programme laufen in einer eigenen Sandbox
- Keine Common Storage Area
- Zugriff auf Daten oder Dateien anderer Anwendungen
- Durch den Content Provider werden Entwickler angehalten, Schnittstellen zu definieren.
- oder vorgegebene zu benutzen

# Warum Content Provider?

Funktionsweise:



# Beispiele für existierende Content Provider

(aus dem Packages android.provider.*)

- Contacts
  - Adressbuchdaten
- MediaStore
  - Multimediadaten
- CallLog
  - Anruferliste
- Settings
  - Systemeinstellungen

# Prinzip der Content Provider

- Die Ansprache:
  - Universal Resource Identifier (URI)
- Die Nutzung:
  - Lesen und Verändern von Daten

# Prinzip der Content Provider

Zugriff auf einen Content Provider erfolgt mittels URI

Struktur:

**Schema://Name/Beschreibung/ID**

# Prinzip der Content Provider

Aufbau des URI:

**Schema://**Name/Beschreibung/ID

- gibt die Art der Datenquelle an:
  - android.resource://
    - Android-Umgebung
  - file://
    - Datei
  - content://
    - Datenbank- oder Binärinhalt

# Prinzip der Content Provider

Aufbau des URI:

Schema://**Name/**Beschreibung/ID

- gibt den vollqualifizierten Namen der implementierten Klasse an.

  Beispiel:

  - browser/
  - contacts/

# Prinzip der Content Provider

Aufbau des URI:

Schema://Name/**Beschreibung/**ID

- gibt den Pfad der zurückzugebenden Informationen an.

  Beispiel:

  - people/
    - gibt die Personen aus der Anruferliste zurück

# Prinzip der Content Provider

Aufbau des URI:

Schema://Name/Beschreibung/**ID**

- beschränkt die Daten auf bestimmte Schlüsselwerte (optional)

# Prinzip der Content Provider

URI:

- URI können über die Dokumentation herausgefunden werden
- Der URI liefert i. d. R. einen Cursor auf die Zieldatenmenge
- Wenn die Feldnamen bekannt sind, kann gezielt darauf zugegriffen werden
- Wenn nicht, ist das nur durch gezieltes Debuggen möglich

Die fertige URI wird in der Manifest.xml eingetragen:

```
<provider name="com.example.exampleProvider"
        authorities="com.example.exampleprovider"
        . . . />
</provider>
```

# Prinzip der Content Provider

- Beispiel Telefonbuch:

```
import android.provider.Contacts.People;
import android.database.Cursor;


// Form an array specifying which columns to return.
String[] projection = new String[] {
                      People._ID,
                      People._COUNT,
                      People.NAME,
                      People.NUMBER
                     };
// Get the base URI for the People table in the Contacts content
provider.
Uri contacts = People.CONTENT_URI;

// Make the query.
Cursor managedCursor = managedQuery(contacts,
                projection, // Which columns to return
                null, // Which rows to return (all rows)
                null, // Selection arguments (none)
                // Put the results in ascending order by name
                People.NAME + " ASC");
```

# Prinzip der Content Provider

Datenbestand:

| _ID | NUMBER | NUMBER_KEY | LABEL | NAME | TYPE |
|---|---|---|---|---|---|
| 13 | (425) 555 6677 | 425 555 6677 | Kirkland office | Bully Pulpit | TYPE_WORK |
| 44 | (212) 555-1234 | 212 555 1234 | NY apartment | Alan Vain | TYPE_HOME |
| 45 | (212) 555-6657 | 212 555 6657 | Downtown office | Alan Vain | TYPE_MOBILE |
| 53 | 201.555.4433 | 201 555 4433 | Love Nest | Rex Cars | TYPE_HOME |

Rückgabe:

| _ID | _COUNT | NAME | NUMBER |
|---|---|---|---|
| 44 | 3 | Alan Vain | 212 555 1234 |
| 13 | 3 | Bully Pulpit | 425 555 6677 |
| 53 | 3 | Rex Cars | 201 555 4433 |

# Prinzip der Content Provider

- Beispiel für Veränderung des Bestandes:

```
import android.provider.Contacts.People;
import android.content.ContentResolver;
import android.content.ContentValues;


ContentValues values = new ContentValues();

// Add Abraham Lincoln to contacts and make him a
favorite.
values.put(People.NAME, "Abraham Lincoln");

// 1 = the new contact is added to favorites
// 0 = the new contact is not added to favorites
values.put(People.STARRED, 1);

Uri uri = getContentResolver().insert(People.CONTENT_URI,
values);
```

# Sicherheit

- Exkurs: Sicherheitsstruktur von Android
- Zugriffsrechte
- URI Permission

# Exkurs: Sicherheitsstruktur

- Es gibt ein Rechtesystem
- Jede Anwendung läuft in einer eigenen Sandbox
- Anwendungen haben standardmäßig nicht das Recht, sich gegenseitig zu beeinflussen
- Jede Anwendung muss signiert werden (allerdings nicht von einer offiziellen Stelle)

# Sicherheit

- Zugriffsrechte:
  - Jede Anwendung erhält ihre individuelle User-ID
  - Diese ID ist lebenslang gültig
  - Jede Anwendung läuft unter dieser ID in einem Sandbox-Prozess
  - Die ID gibt an, wo die Daten gespeichert werden
  - Die einzige Ausnahme: Zwei Anwendungen haben in ihrer Manifest.xml ein sharedUserId-Flag gesetzt
  - Ein Anwendung muss in ihrer Manifest.xml anzeigen, welche Rechte sie anfordert

# Sicherheit

- URI Permission
  - Die Zugriffsrechte auf Content Provider werden analog der Zugriffsrechte in der Manifest.xml verwaltet

Eintrag in der Manifest.xml:

```
<grant-uri-permission android:path="string"
                      android:pathPattern="string"
                      android:pathPrefix="string" />
```

# Vor- und Nachteile von Content Providern

- Vorteile:
  - Universelle Schnittstellen
  - Vorgegebene Struktur
  - Sicherheitskonzept

- Nachteile:
  - Übermittelt „nur“ einen Cursor
  - Es müssen immer alle Methoden für die DB-Abfragen implementiert sein

# Alternativen zum Content Provider

- Nutzung der java.io
  - Funktioniert nur bei SD-Karten
  - Funktioniert nur auf Dateisystemebene
  - Unterwandert das Sicherheitsprinzip
- Nutzung eines Remote Service
  - Bei kleinen Datenmengen performanter
  - Bei großen unperformant
  - Für kleine Mengen komplexer Datenstrukturen gut

# Inhalt

1. Überblick Schnittstellen
   - Welche Schnittstellen gibt es?
   - Was ist das besondere an Schnittstellen bei Android?
2. Content Provider
   - Warum Content Provider
   - Beispiele für existierende Content Provider
   - Funktionsprinzip der Content Provider
   - Sicherheit
   - Vor- und Nachteile
   - Alternativen zum Content Provider
3. Netzwerk-Schnittstellen
   - Besonderheiten von Netzwerken bei mobilen Geräten
   - Welche Schnittstellen hält Android bereit?
   - Funktionsweise an einem Beispiel

# Besonderheiten von Netzwerken bei mob. Geräten

- Variable Signalqualität
- Variable Datenrate
- Netzwerkzugang kann sich ändern
- Uneinheitliche Kostenstruktur bei den einzelnen Netzwerkverbindungen

# Netzwerk-Schnittstellen

- Welche Schnittstellen hält Android bereit?
  - java.net.*
  - org.apache.http
  - android.net.*

# Welche Schnittstellen hält Android bereit?

- java.net.* (Auszüge)
  - ContentHandler
  - DatagramSocket
  - InetAddress
  - NetworkInterface
  - SocketAdress
  - URL

# Welche Schnittstellen hält Android bereit?

- Jakarta Commons HttpComponents (org.apache.http)
  - bietet diverse Features für http, die nicht Bestandteil von java.net sind
    - verbesserte Cookie-Unterstützung
    - verbessertes SSL-Handling
    - http GET/POST Implementierung
    - ...

# Welche Schnittstellen hält Android bereit?

- Jakarte Commons HttpComponents
  - unterteilt sich in zwei Teile
    - HttpComponents Core
      - Implementierung der fundamentalen HTTP-Konzepte
    - HttpComponents Client
      - Clientseitige Implementation von HTTP-Konzepten

# Welche Schnittstellen hält Android bereit?

- android.net
  - ConnectivityManager
    - Benachrichtigung bei Verbindungsabruch
    - Automatisches Failover
  - NetworkInfo
    - Informationen über die Netzwerkverbindung

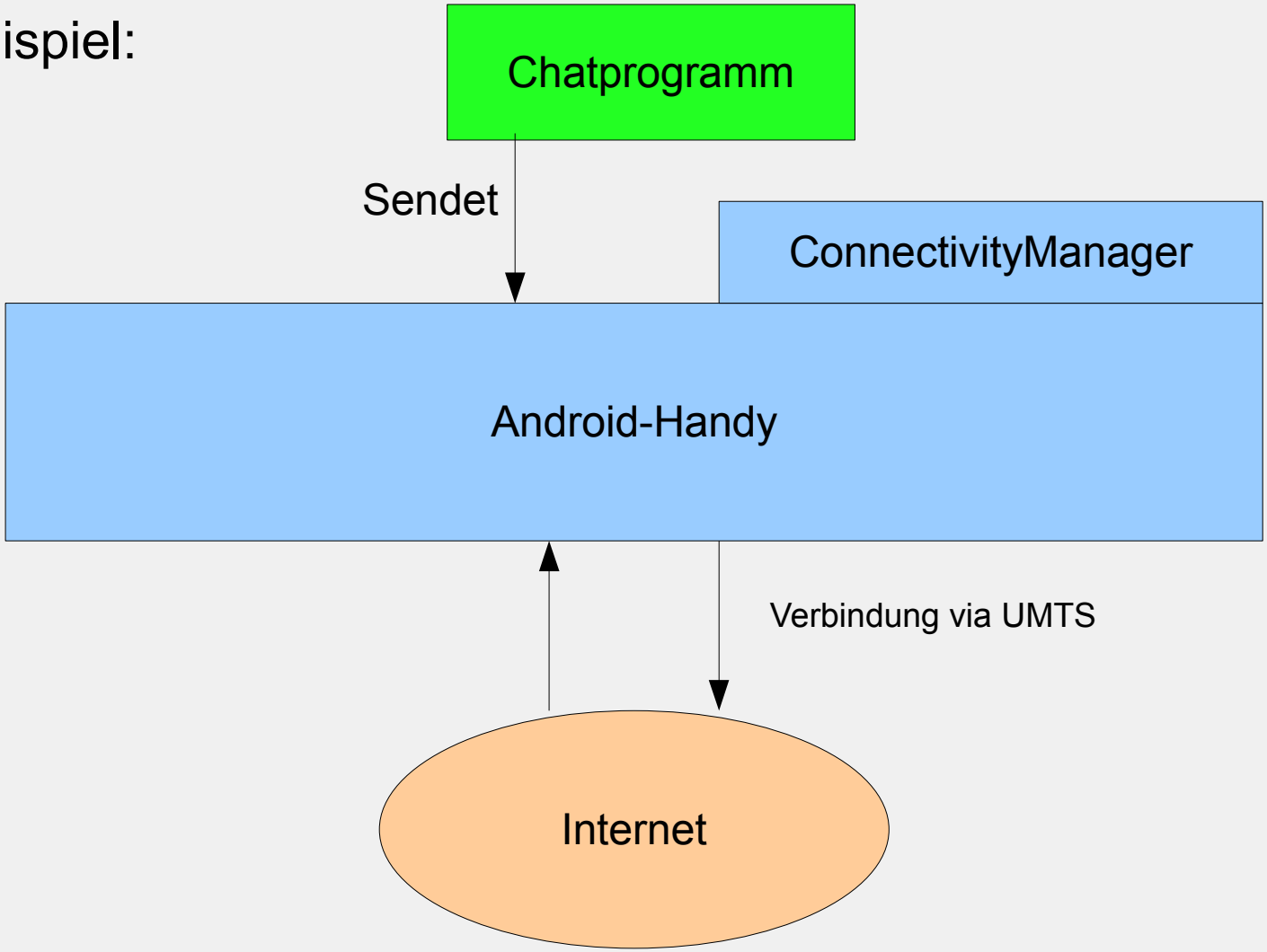
# Netzwerk-Schnittstellen

- ConnectivityManager
  - Hintergrundservice
  - Sendet Broadcast Intent bei Verbindungsabbruch.
    Dieser kann durch einen Broadcast Receiver in der Anwendung aufgefangen und verarbeitet werden.
  - Sendet ebenfalls einen, bei wieder aufgenommener Verbindung
  - Verwaltet alle Netzwerkverbindungen. Bei Wegfall einer Verbindung versucht er automatisch eine neue aufzubauen.
  - wird erzeugt durch:

```
ConnectivityManager cm =
        Context.getSystemService(
                Context.CONNECTIVITY_SERVICE)
```

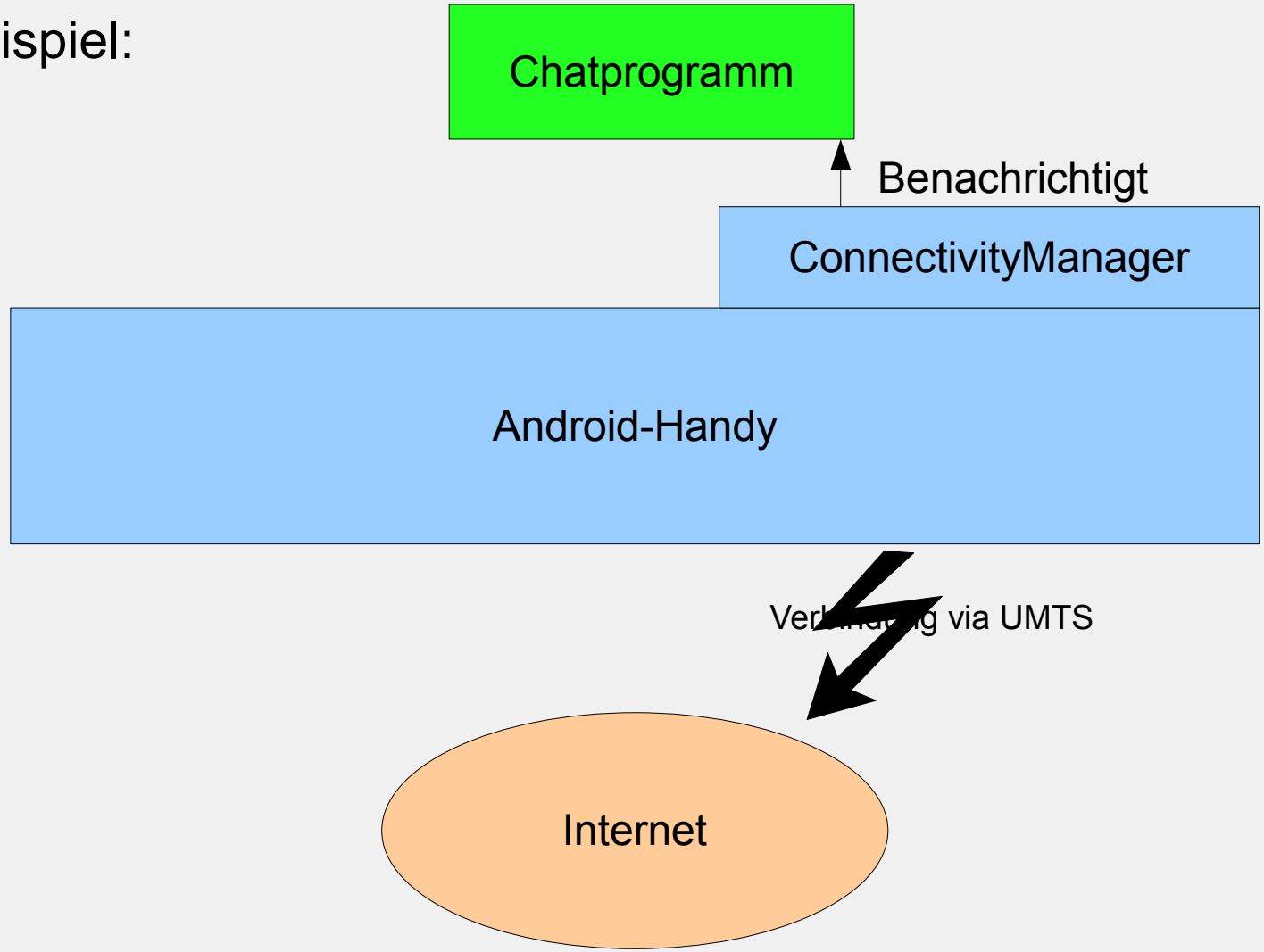
# Netzwerk-Schnittstellen

- Beispiel:



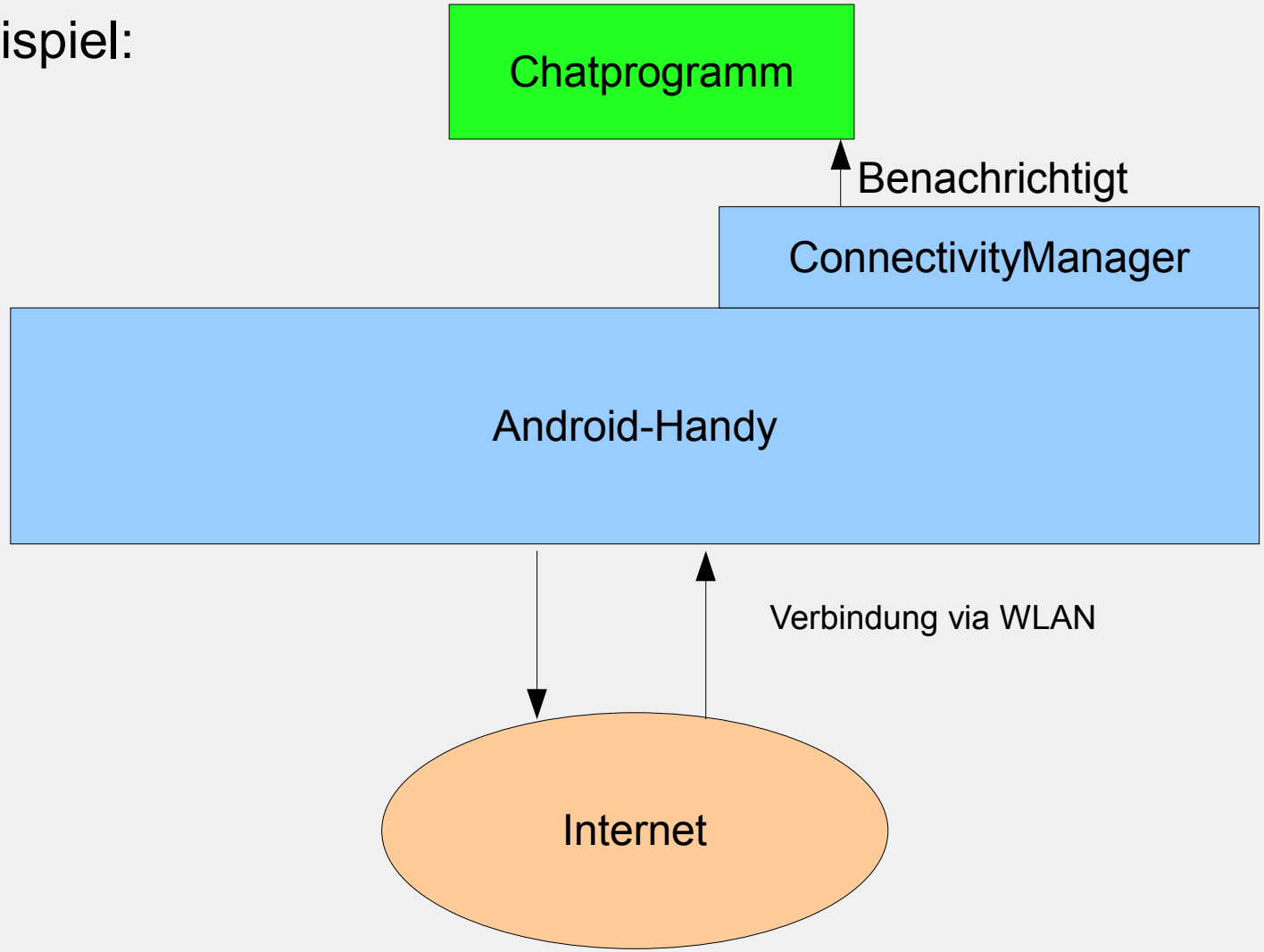
# Netzwerk-Schnittstellen

- Beispiel:



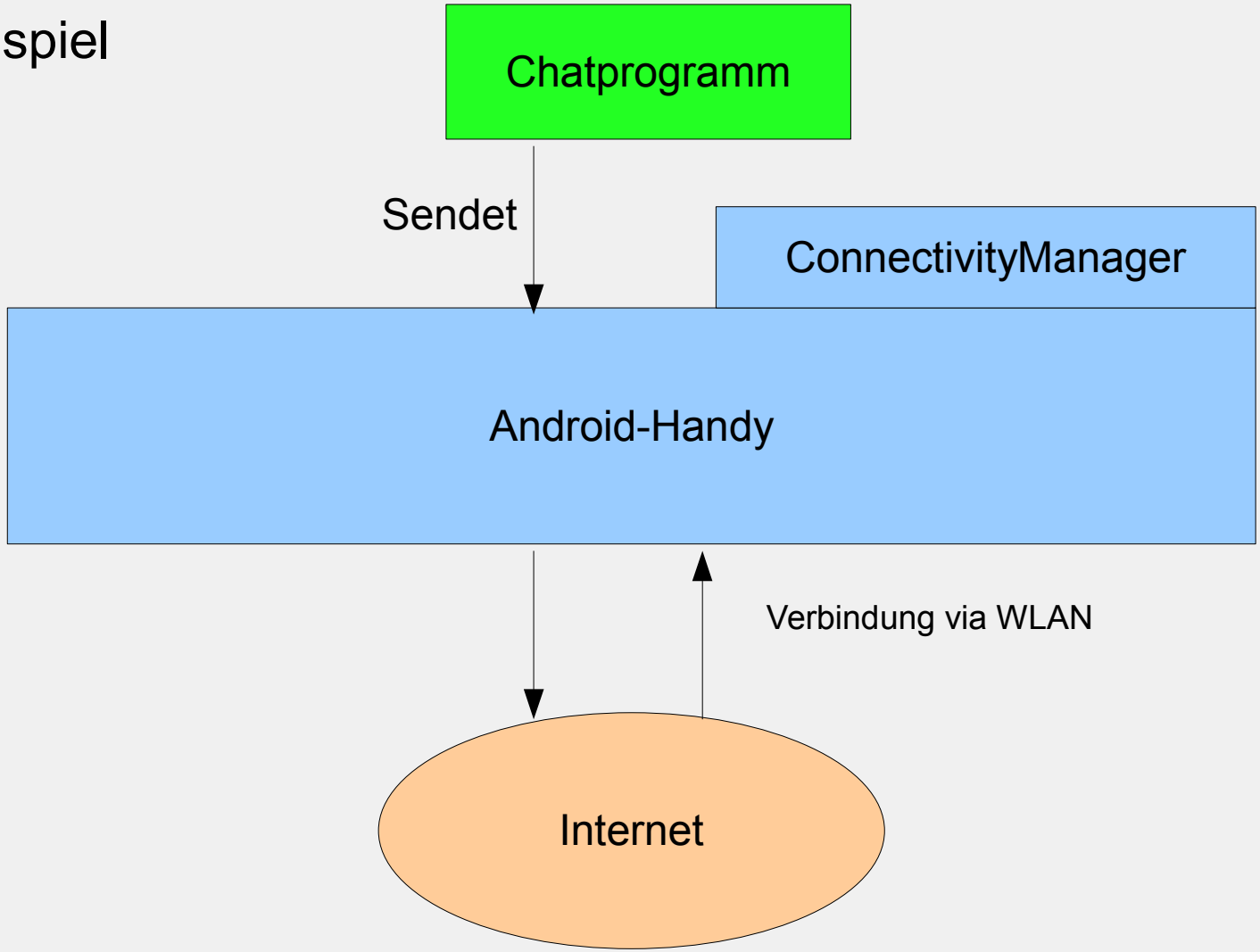
# Netzwerk-Schnittstellen

- Beispiel:



# Netzwerk-Schnittstellen

- Beispiel



# Netzwerk-Schnittstellen

- NetworkInfo:
  - Datentyp für Netzwerkverbindungen
  - wird durch Methoden am ConnectivityManager erzeugt
    - getActiveNetworkInfo()
    - getAllNetworkInfo()
    - getNetworkInfo(int networkType)
  - liefert Informationen über das Netzwerk
    - getState()
    - isConnected()
    - getType()
    - getTypeName()

# Netzwerk-Schnittstellen

- android.net.wifi
  - Android bietet Schnittstellen speziell für WLAN.
    - WifiManager
    - WifiManager.MulticastLock
    - WifiManager.WifiLock
    - ...

# Netzwerk-Schnittstellen

- Wie läuft das ab?

  Folgendes Szenario:

  - Der Android-Benutzer hat einen Instant-Messenger
  - Er zahlt über GPRS nach Datenmenge
  - Bei WLAN zahlt er nichts
  - Er holt die Nachrichten vom Server
  - Je nach Verbindung soll die Abholfrequenz angepasst werden

# Netzwerk-Schnittstellen

- Code:

```
ConnectivityManager cm =
     Context.getSystemService(
          Context.CONNECTIVITY_SERVICE);
...
if (arbeit2 == null) //damit maximal ein task erstellt wird und nicht jedes mal ein
neuer.
{
     if(cm.getActiveNetworkInfo().getType()==0)
     {
          time = 1000;
     }
     else{
          time = 5000;
          }
     arbeit2 = new arbeit();
     timer.schedule(arbeit2, 1,time);
}
...
private class arbeit extends TimerTask{
     public void run()
     {
          msgOut[] verarbeitung = liste.listeVersenden();
          String sendung = "";
          sendung += new msgOutPresence(_user,_pass,_status,_status2).Get_String();

     for(int i=0;i<verarbeitung.length;i++)
     {
          sendung += verarbeitung[i].Get_String();
     }
          eingang(server.Pull(sendung));
     }
}
```

# Fazit:

- Speicher:
  - Klar strukturierte Schnittstellen
  - Berücksichtigung einer Sicherheitsarchitektur auf OS-Ebene
  - Möglichkeit freier Nutzung eigener Software
- Netzwerk:
  - Vereinfachte Schnittstelle
  - Failover-System

# Literatur

- http://developer.android.com/index.html
  - Die offizielle Homepage
- Android - Grundlagen der Programmierung
  - Arno Becker, Marcus Pant

# Das war's...

- Vielen Dank für's Zuhören!