Android Context Twitter

Software Engineering Projekt WS 09/10

Gliederung

- Motivation
- Zielsetzung
- Grundlagen
- Implementation
  - Komponentenüberblick
  - Komponentenkommunikation
  - Serverkomponenten
  - Android Context Twitter (ACT)
- Ergebnisse / Zusammenfassung

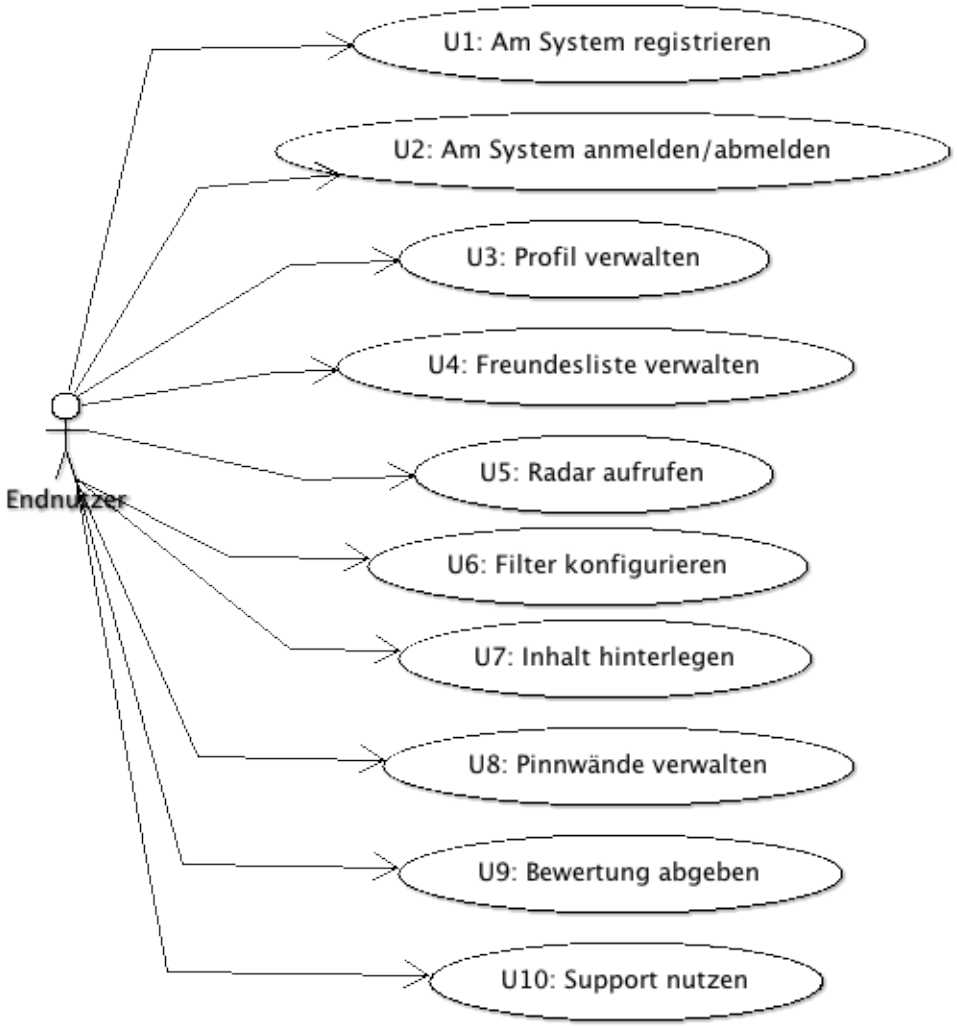
Motivation

- Zunehmende Verbreitung von Smartphones, auch mit GPS Modulen
- Erhöhte Mobilitätsanforderungen der Benutzer
- Viele und schnell wachsende Social-Network-Communities
- Fortlaufende Veröffentlichung und Zentralisierung von Nutzerdaten und –content
- Nahezu alle Social Networks mit eigenen Applikationen auf Handys vertreten
- Neue Visualisierungsmöglichkeiten dank umfangreicher Sensorenausstattung (GPS, Kompass, Motion, Kamera) möglich >> „Augmented Reality“

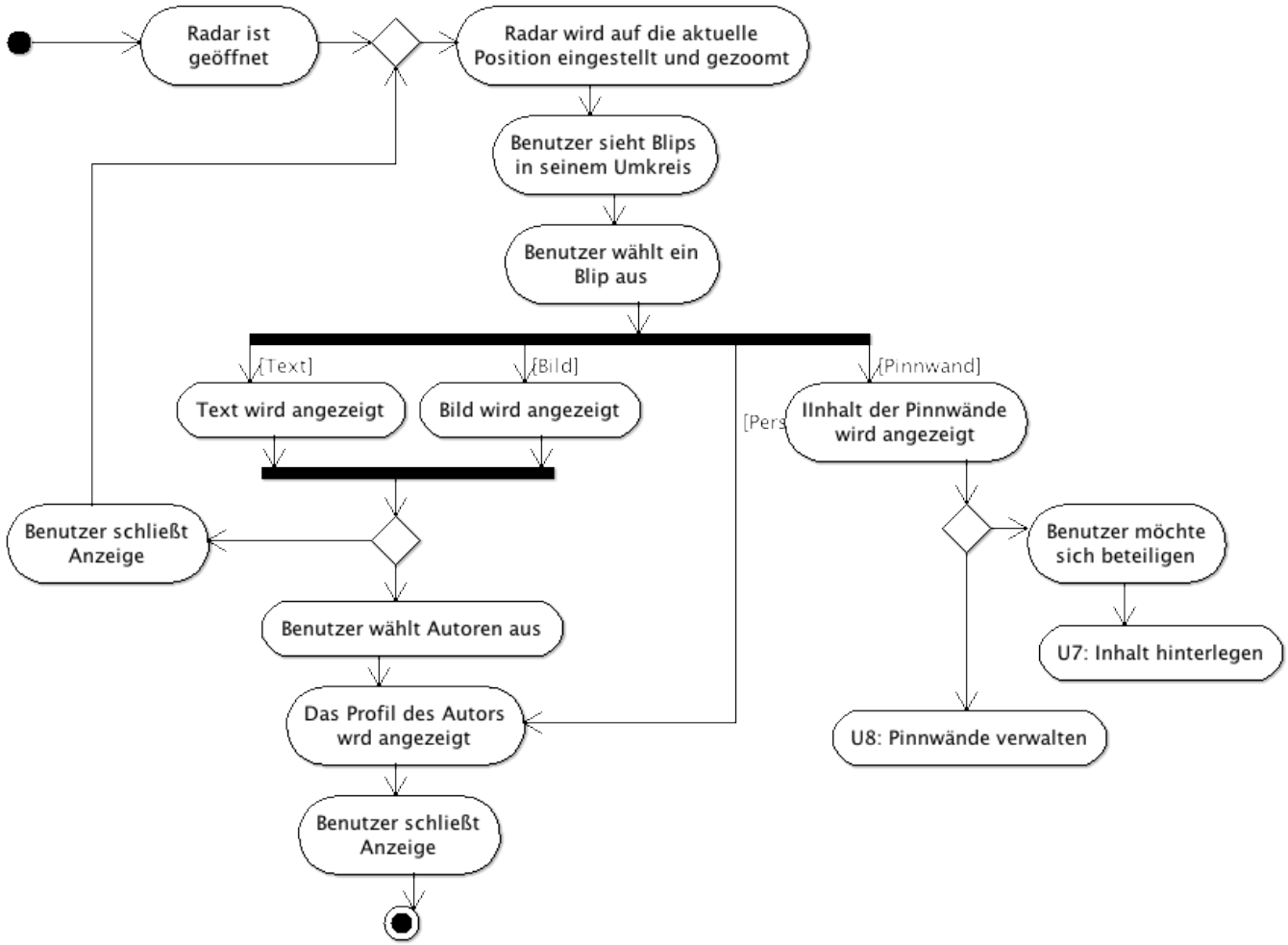
Grobe Zielsetzung

- Umsetzung eines prototypischen handybasierten Social Network Dienstes:
  - Ablage und Veröffentlichung ortsgebundener Inhalte durch Benutzer
  - Ortsgebunden durch Nutzung von GPS und der Google Maps API
  - Sinnvolle Verknüpfung des Benutzerprofils mit extern bereitgestellten Informationen des Context Brokers
- Diese Vorlage wurde von uns durch umfangreiche, weitere Zielsetzungen erweitert, von denen wir uns auf die Wichtigsten konzentriert haben

Spezifizierte Anwendungsfälle



Exemplarisches Aktivitätsdiagramm



Grundlagen

- PHP
- XML
- Android SDK

Einführung in PHP

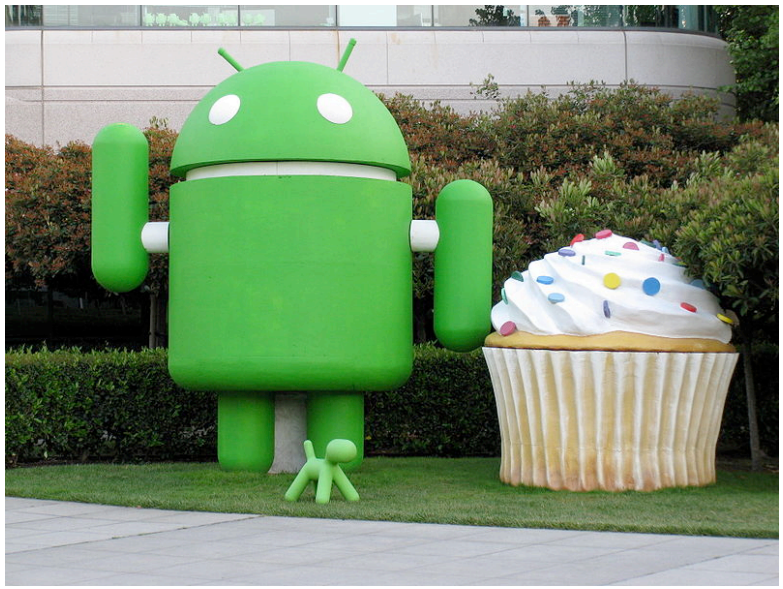
- PHP Hypertext Preprocessor
- Serverseitige Skriptsprache
- Objektorientiert
- Syntax ähnlich Java, C, Perl
- Einfach zu erlernen und zu handhaben
- Schnelle Ergebnisse
- Gut dokumentiert, große Community, viele hilfreiche Webseiten
- Keine gravierenden Nachteile gegenüber Java
- Gute Kenntnisse im Team vorhanden

XML

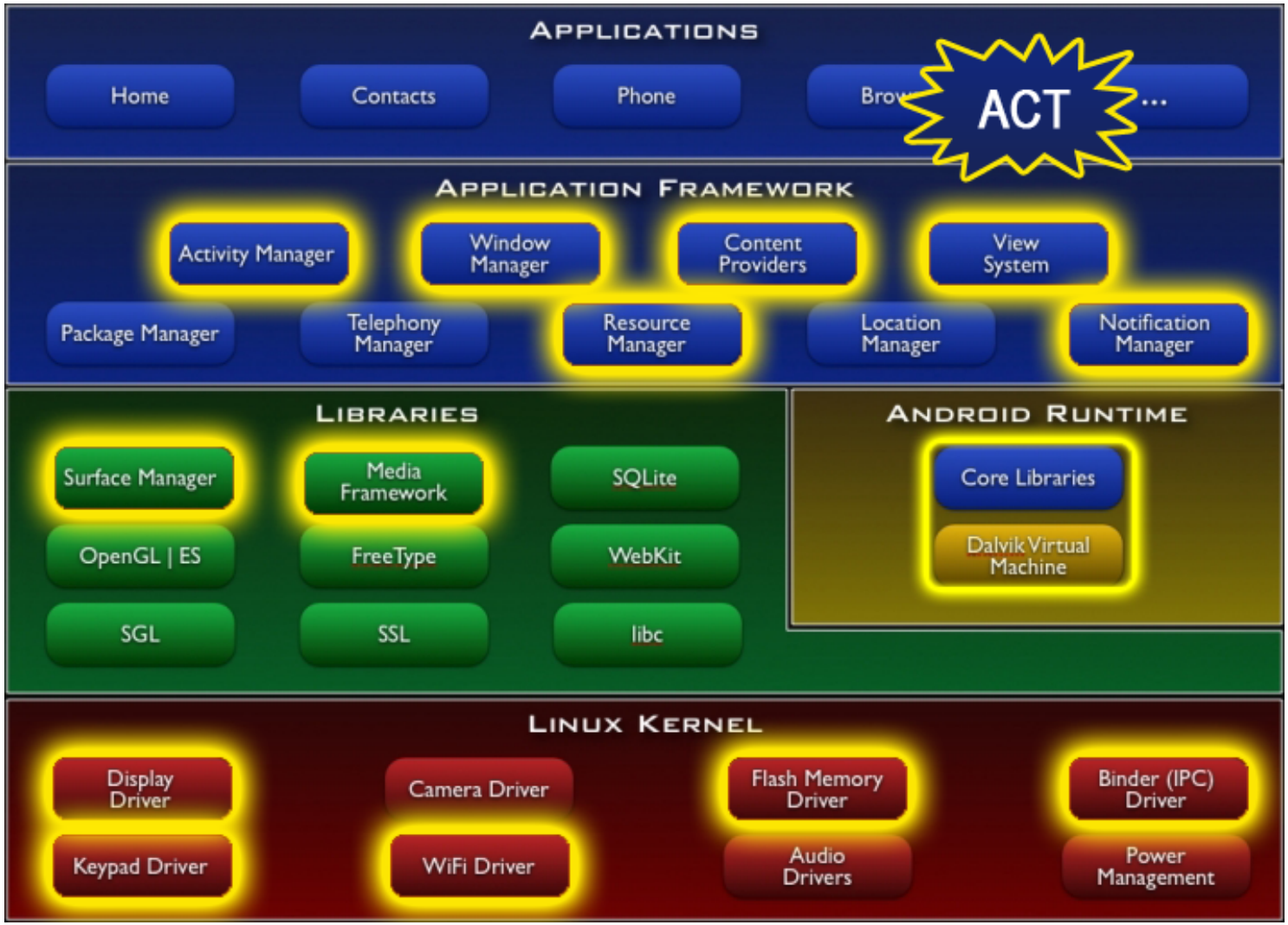
- eXtensible Markup Language
- Context Broker verwendet XML
- Zur Übertragung von komplexen Daten
- Aufruf von Funktionen mit komplexen Daten
- Erweiterbar und Wiederverwendbar
- XML-Parsing in den wichtigsten Programmiersprachen schon implementiert

Android SDK

- Android OS Release: Oktober 2008
- Baut auf Linux-Kernel auf
- Hat effiziente Java VM
- 511 von 1448 Java-Klassen sind Android-spezifisch
- Framework ist stark modular
- Anwendung läuft über Activities
- Strenges Berechtigungssystem
- Einfache Ressourcenverwaltung



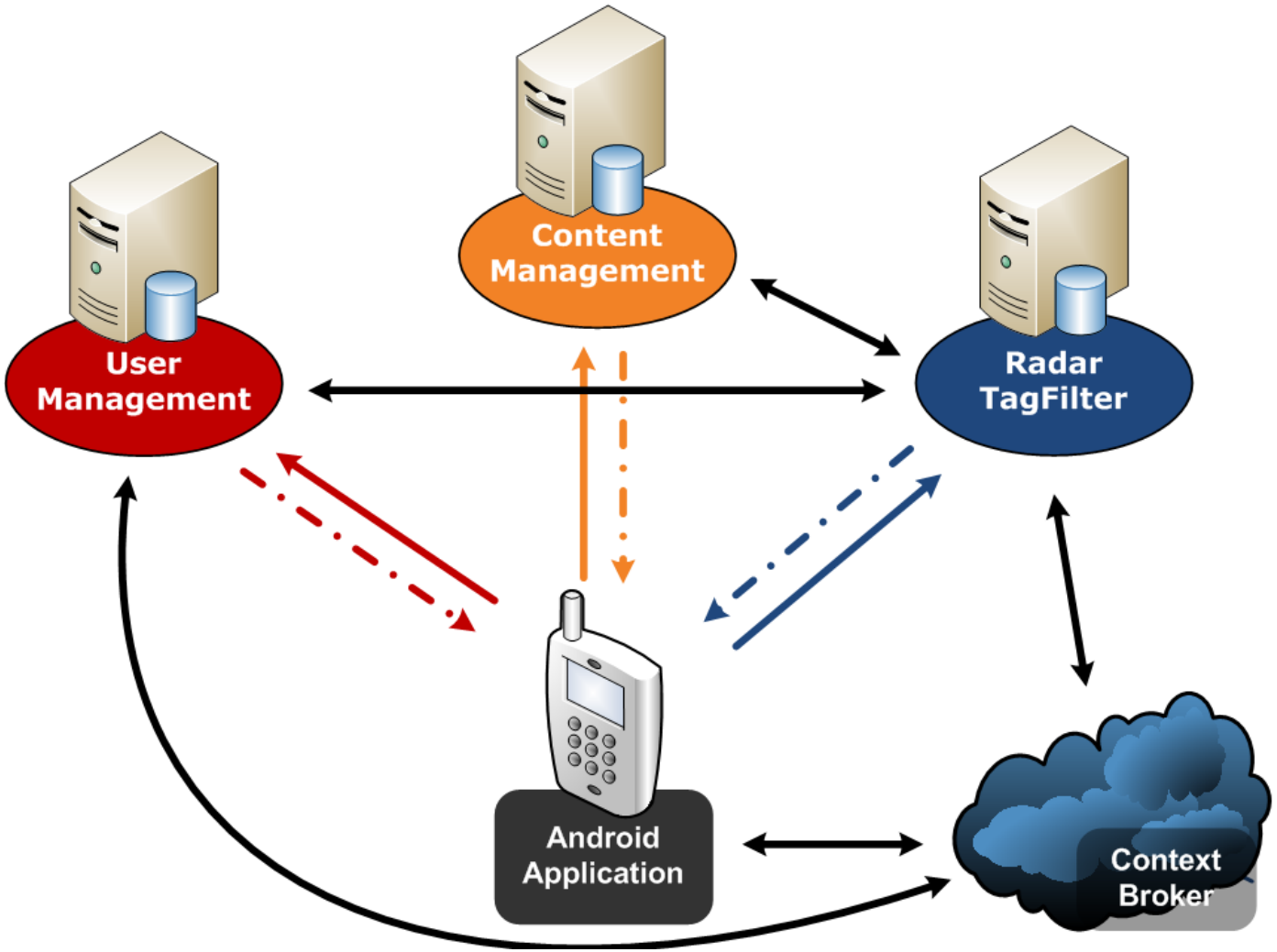
Android SDK


APPLICATIONS
Home
Contacts
Phone
Brow
...
ACT
APPLICATION FRAMEWORK
Activity Manager
Window Manager
Content Providers
View System
Package Manager
Telephony Manager
Resource Manager
Location Manager
Notification Manager
LIBRARIES
Surface Manager
Media Framework
SQLite
OpenGL | ES
FreeType
WebKit
SGL
SSL
libc
ANDROID RUNTIME
Core Libraries
Dalvik Virtual Machine
LINUX KERNEL
Display Driver
Camera Driver
Flash Memory Driver
Binder (IPC) Driver
Keypad Driver
WiFi Driver
Audio Drivers
Power Management

Design und Implementierung

- Komponentenüberblick
- Kommunikation via HTTP und XML
- Funktionalitäten und Schnittstellen der Komponenten
  - User Management System (UMS)
  - Content Management System (CMS)
  - Radar Management System (Radar)
  - Android Context Twitter (ACT)
    - Architektur
    - Ablauf und Funktionsweise
    - Auswertung der Serverantwort
    - Darstellung der Serverantwort

Komponentenüberblick



Kommunikation via HTML mit XML

- Konventionen:
  - Datenaustauschformat XML
  - Komponenten antworten immer mit XML
  - Fehlerabfragen sind mit Fehlercode versehen

Erfolgreiche Abfrage am CMS

```
<root>
  <data>
    <contenturl>http://e04-server3.et.fh-osnabrueck.de/cms/files/1.txt</contenturl>
  </data>
  <serverstats>
    <servercode>00</servercode>
    <exception>ok</exception>
  </serverstats>
</root>
```

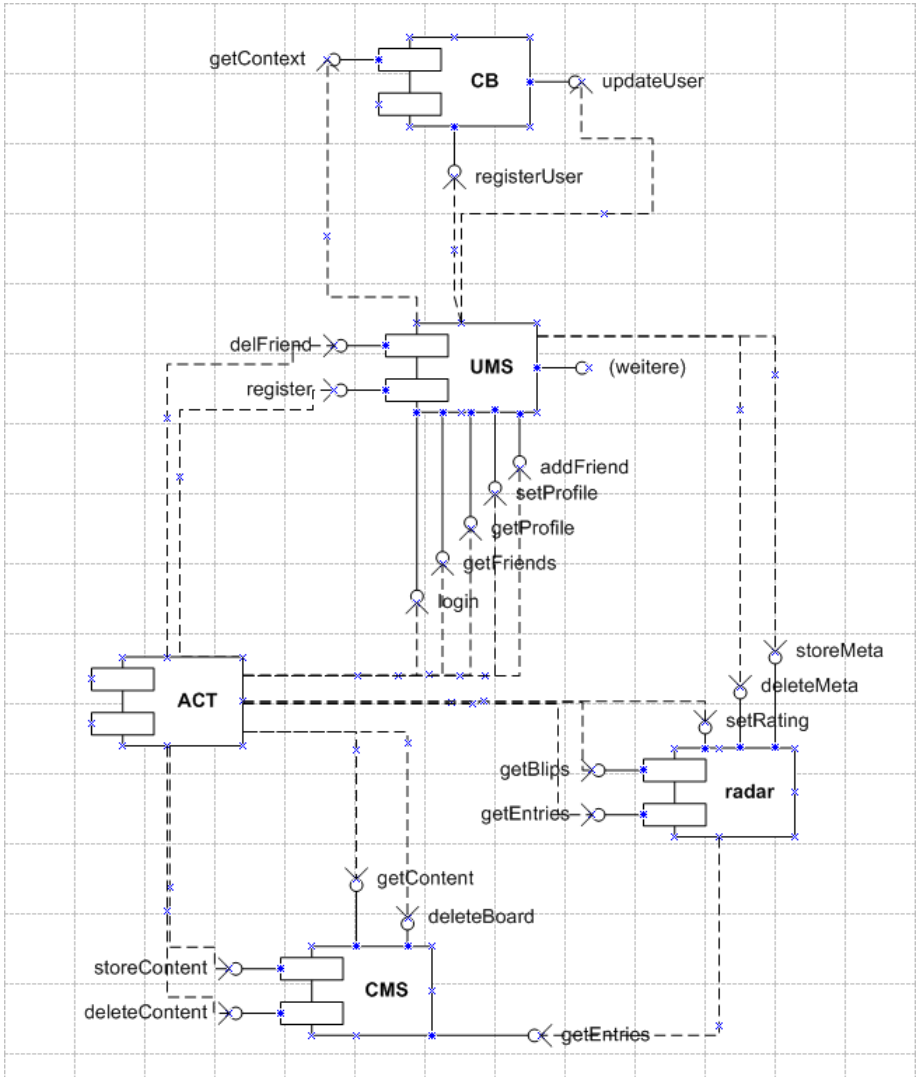
Kommunikation via HTML mit XML

- Konventionen:
  - Datenaustauschformat XML
  - Komponenten antworten immer mit XML
  - Fehlerabfragen sind mit Fehlercode versehen

Erfolglose Abfrage am CMS

```
<root>
  <data/>
  <serverstats>
    <servercode>102</servercode>
    <exception>ContentID ist in der Datenbank nicht vorhanden</exception>
  </serverstats>
</root>
```

Funktionalitäten und Schnittstellen der Komponenten

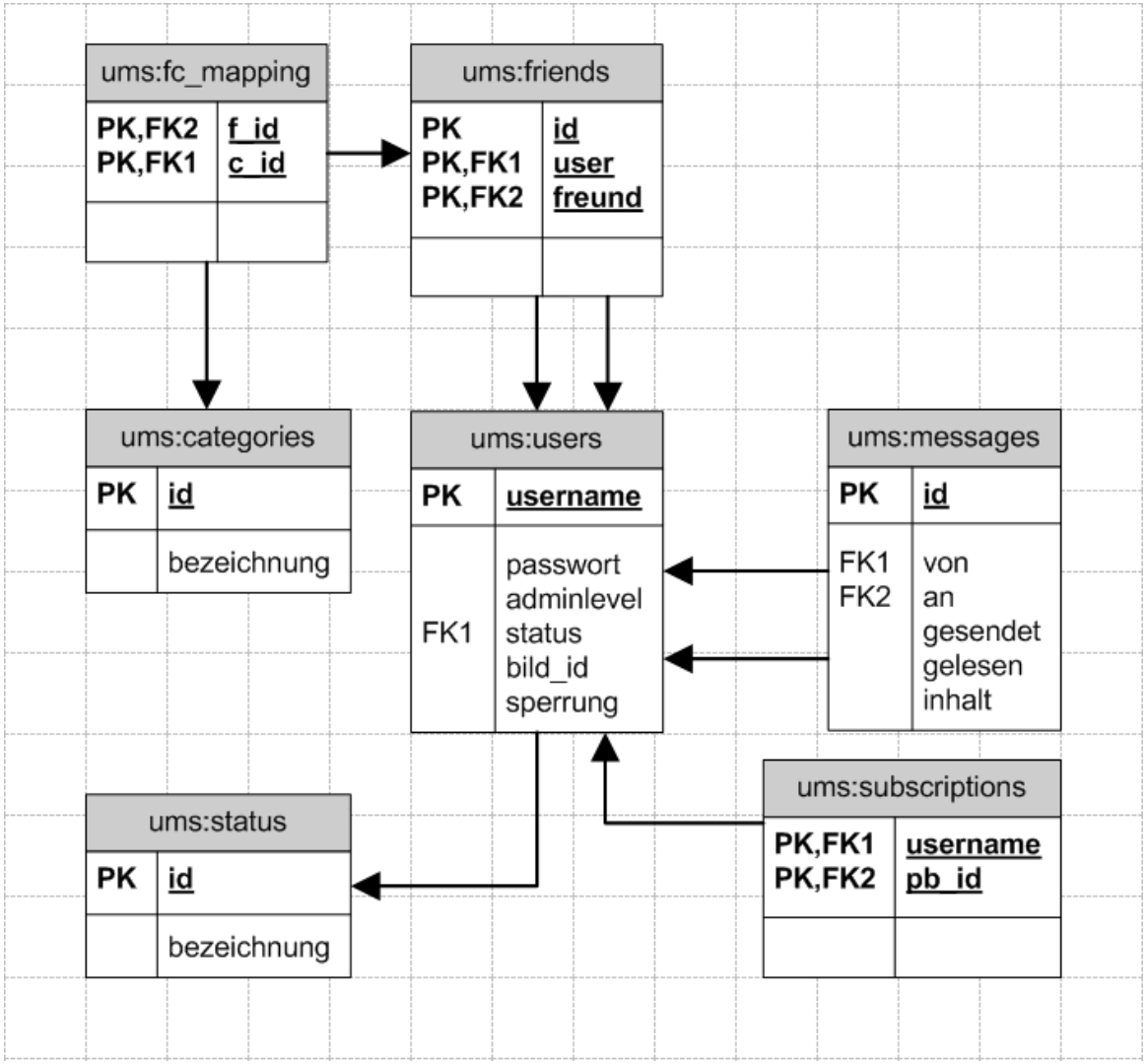


User Management System (UMS)

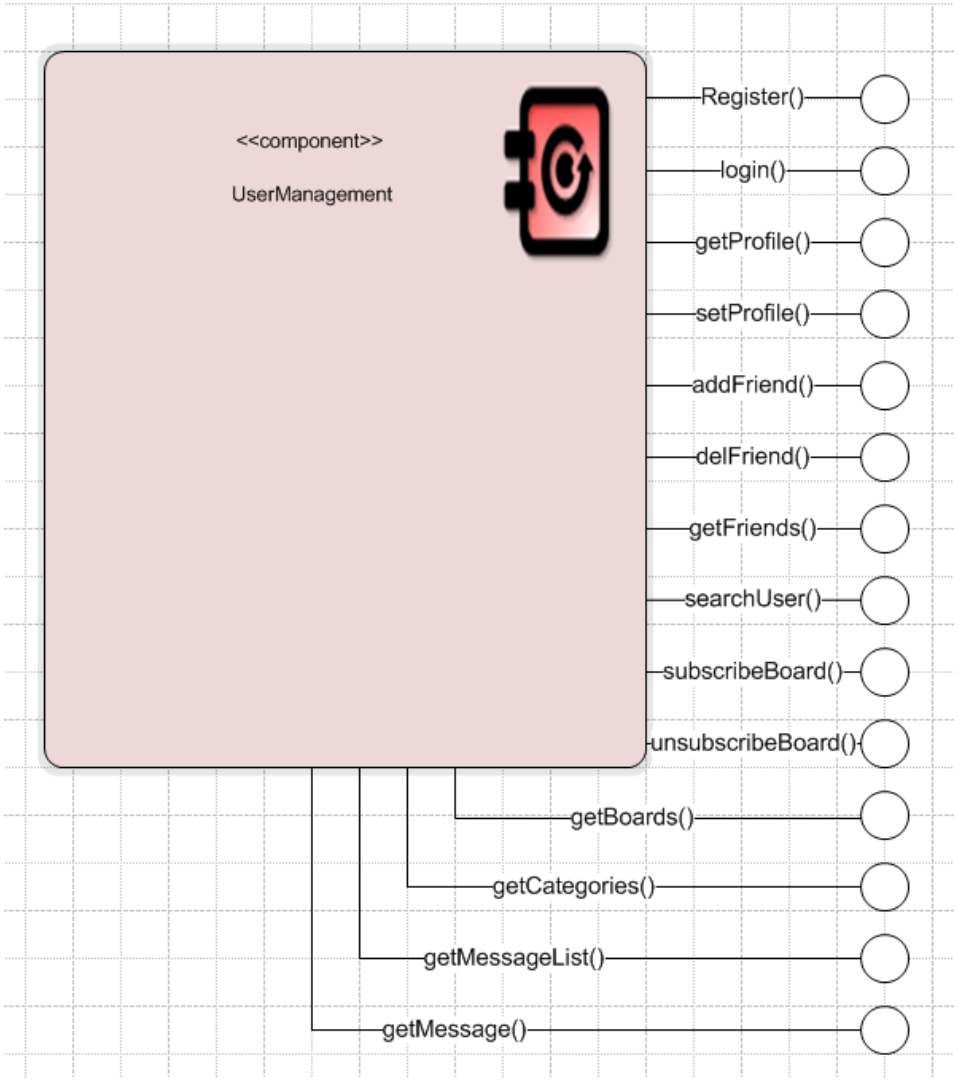
Funktionalität UMS

- Registration von Benutzern
- Profile (vom Context Broker)
  - Lesen
  - Schreiben
- Freundesliste / Funktionen
  - Hinzufügen
  - Entfernen
  - Aufteilung in Kategorien
- Weitere Funktionen, siehe Dokumentation

Datenbank UMS



Komponentendiagramm UMS



Schnittstellen UMS

HTTP (REST) Request, z.B.:

```
/user/getProfile?entity=mknappmeyer
```

Profilfeldändern:

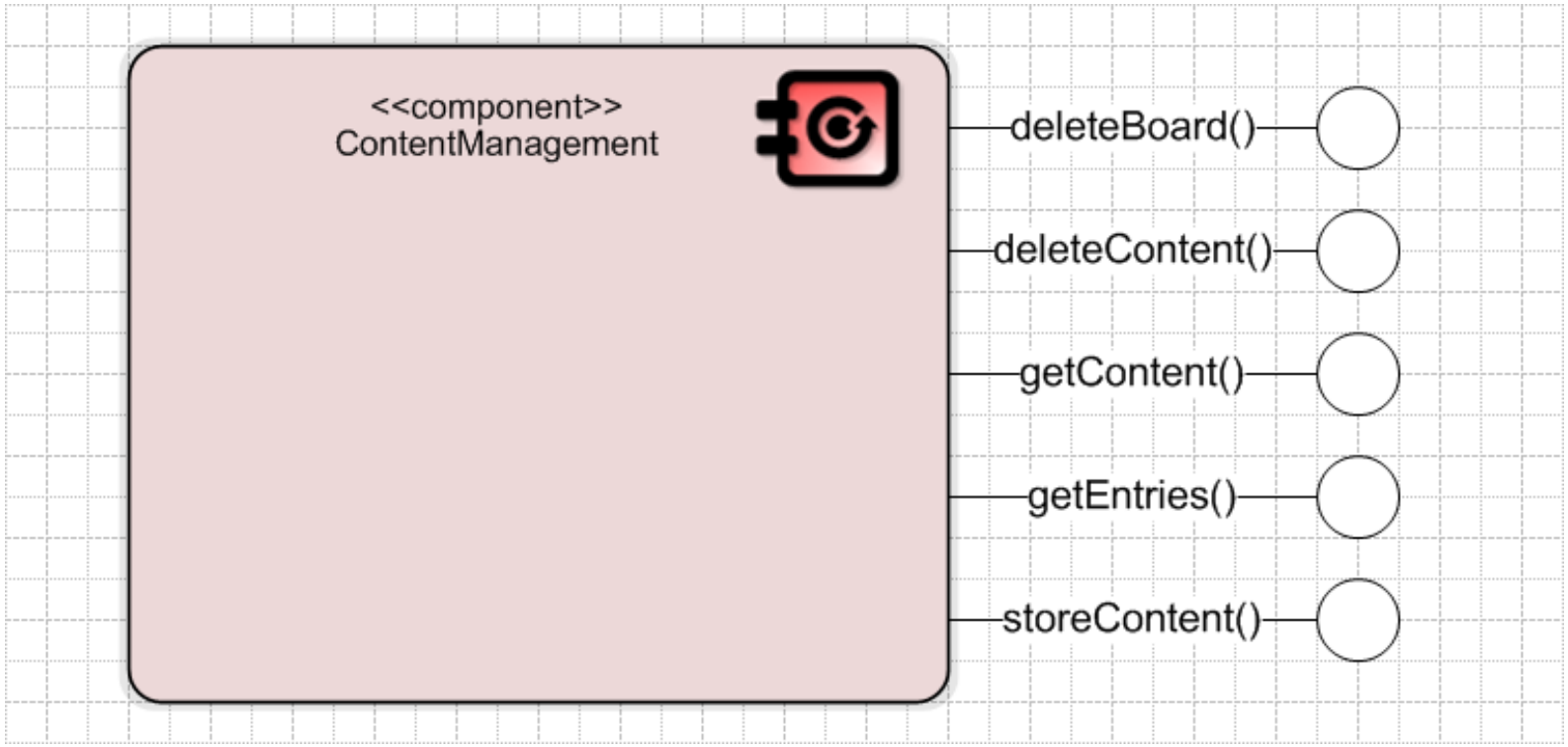
```
/user/setProfile?entity=mknappmeyer&profile=
<data>
  <profile>
    <email>example@example.org</email>
  </profile>
</data>
```

Content Management System (CMS)

Funktionalität CMS

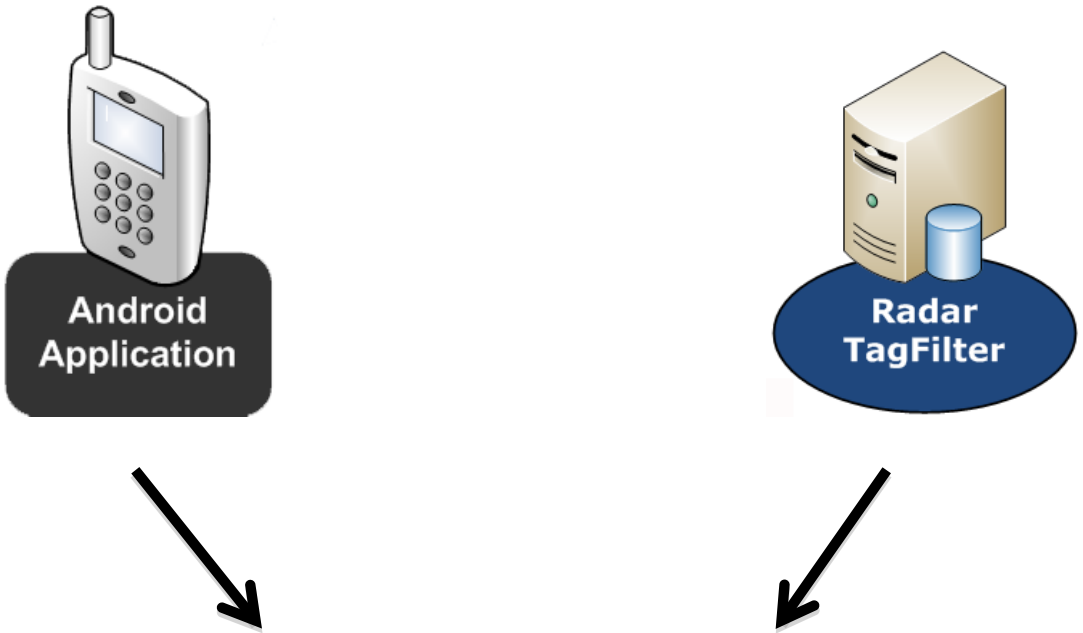
- Serverkomponente
  - Wird angesprochen durch Client und Radar
  - Delegiert Aufgaben an das Radar
  - Referenziert Daten in einer Datenbank
  - Speichert den Inhalt als Dateien im System
  - Kann auf externen Content verweisen
- Content
  - Speichern
  - Löschen
- “Pinnwand”
  - Zuweisung von Contents zu Pinnwänden

Komponentendiagramm CMS



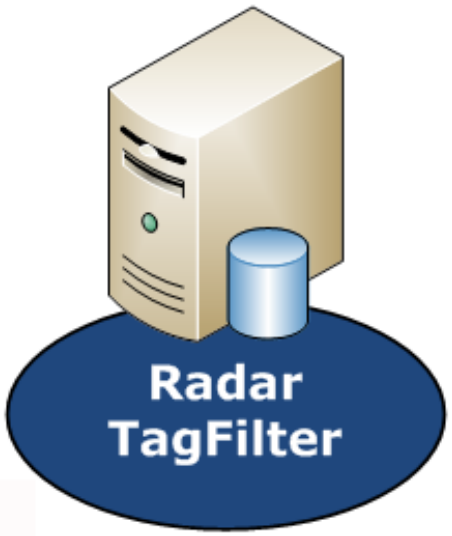
Schnittstellen CMS

Content-Informationenlesen



`/content/getContent?contentid=[id]`

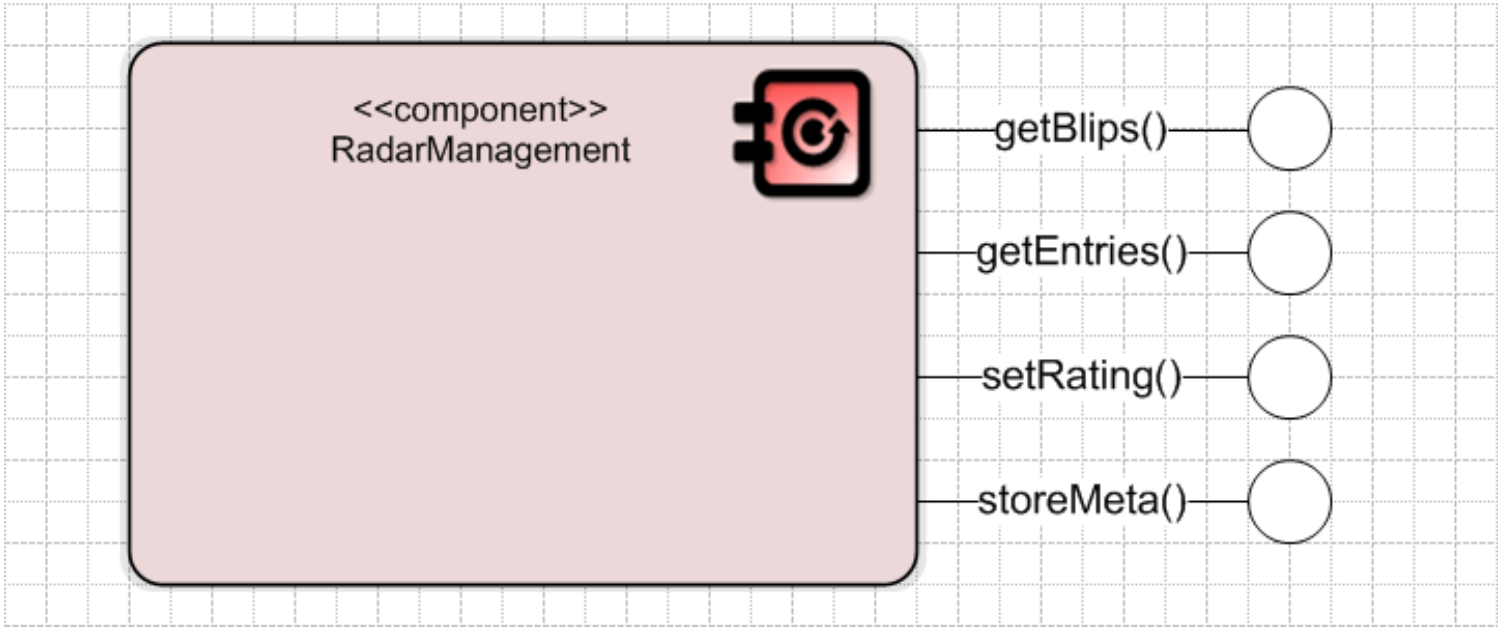
Radar Management System (Radar)



Funktionalität Radar

- Serverkomponente
  - Filtert Content nach Kriterien, wie:
    - Radius in Kilometern um den Benutzer
    - Bewertung
  - Benutzer nach:
    - Geschlecht
    - Alter
  - Liefert „Pinnwände“
  - Für die Contentbewertung zuständig

Komponentendiagramm Radar



getBlips

Aufruf

```
https://e04-server3.et.fh-osnabrueck.de/radar/getBlips?
entity=testy&properties=[XML]
```

XML

```
<data>
  <cat_id>1</cat_id>
  <distance>100</distance>
  <gender>m</gender>
  <userage>40</userage>
  <rating>4</rating>
</data>
```

Radius: Datenbank Abfrage

```
SELECT content.c_id, mediatyp, titel, username, longitude,
   latitude, erstellt
, ACOS(
SIN(RADIANS(latitude)) * SIN(RADIANS(".$lat."))
+ COS(RADIANS(latitude)) * COS(RADIANS(".$lat."))
* COS(RADIANS(longitude) - RADIANS(".$lng."))
) * 6380 AS Distance
FROM locations
INNER JOIN content ON locations.c_id=content.c_id
INNER JOIN viewmodes ON content.c_id=viewmodes.object_id
Having Distance <= ".$saveparams['distance']."
```

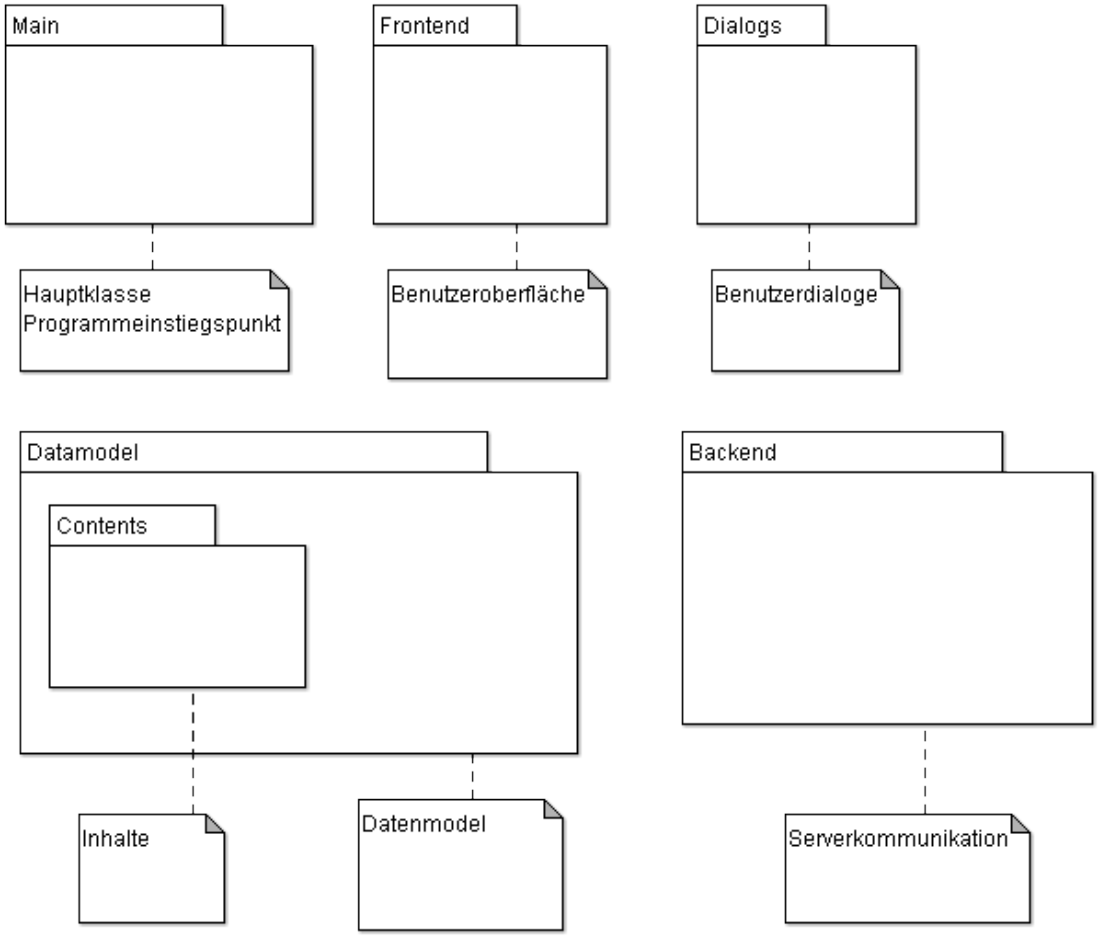
Serverantwort

Antwort

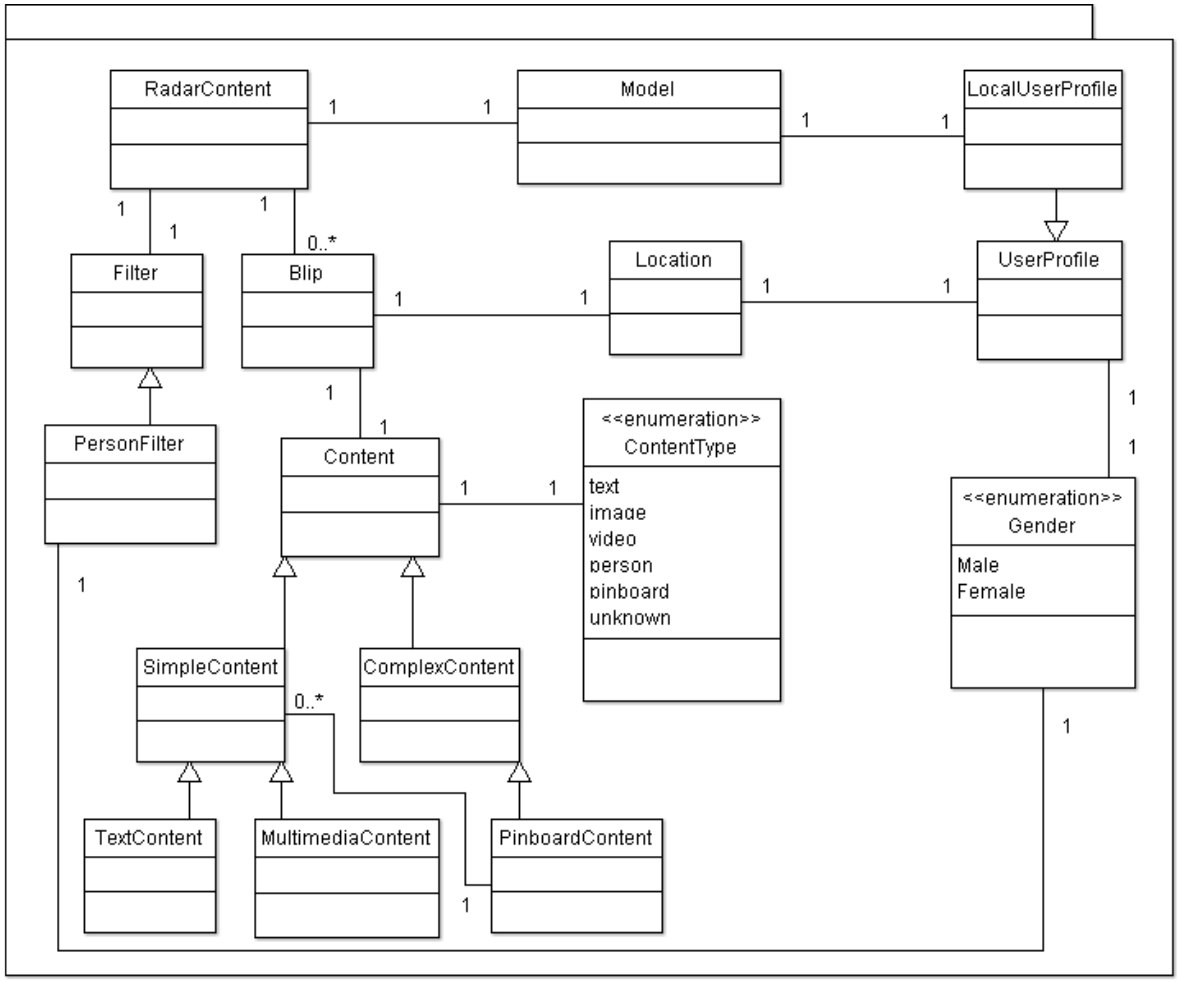
```
<data>
  <blip>
    <longitude>8.05160522460938</longitude>
    <latitude>52.269287109375</latitude>
    <entry>
      …
    </entry>
  </blip>
  <blip>
    …
  </blip>
</data>
```

Android Context Twitter

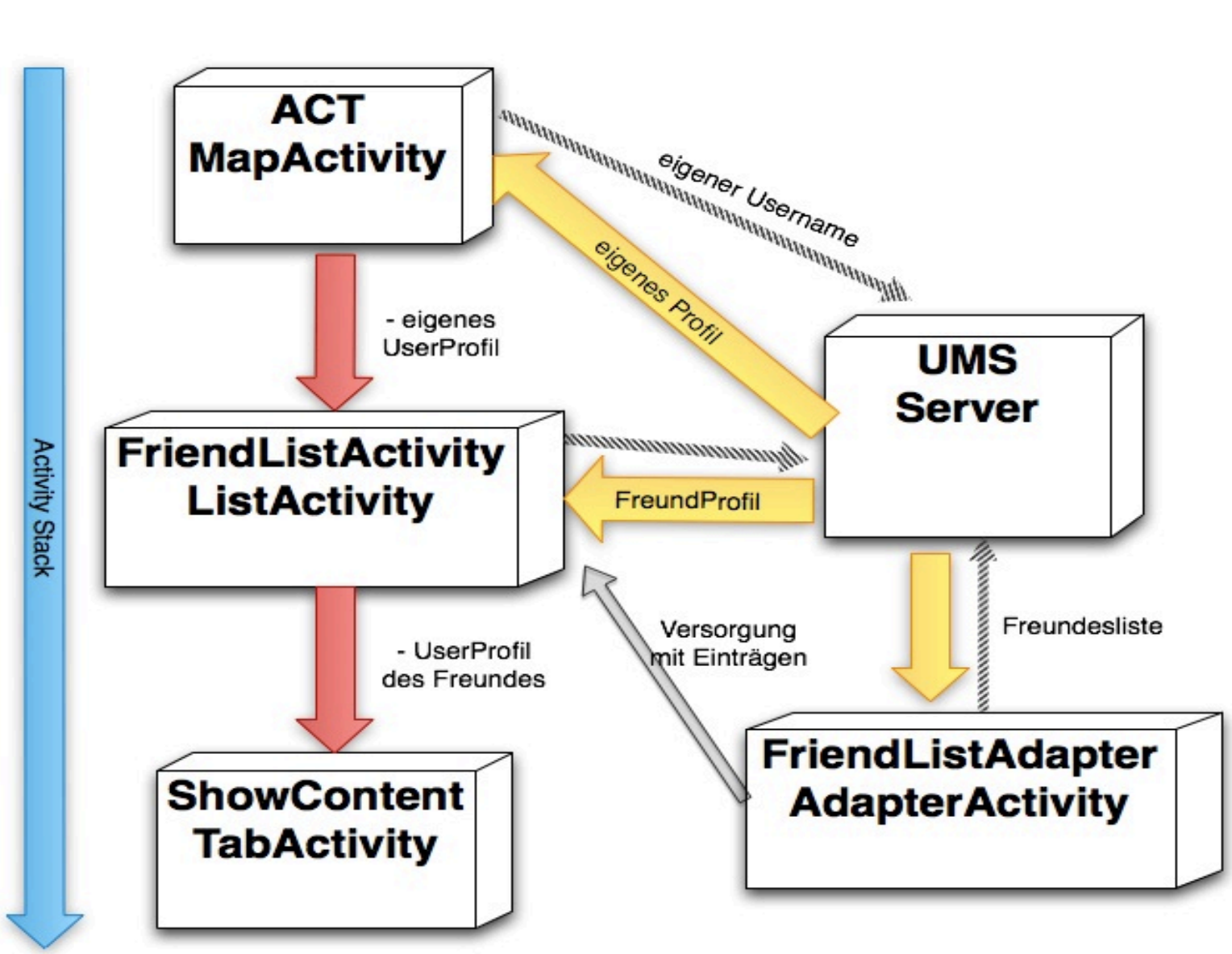
Aufteilung in Pakete

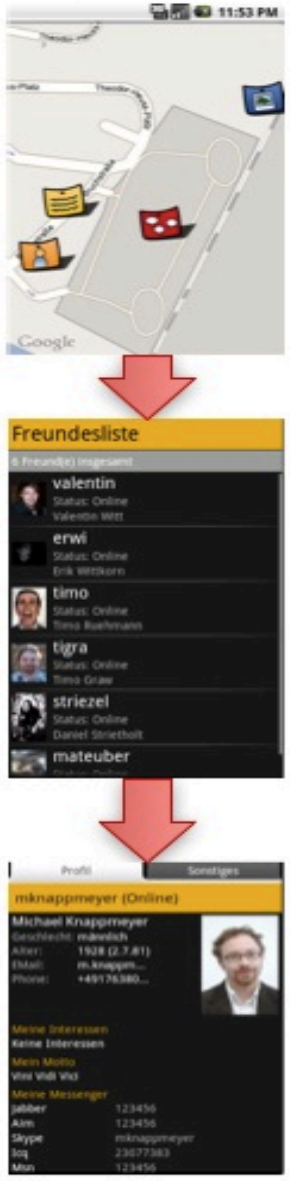


Klassendiagramm Datamodel



Ablaufsteuerung





Auswertung der Serverantwort

- Parsing der XML-Serverantwort
  - XMLContentParser (Basisklasse)
    - parse(InputStream is, String keyword) : Node
  - Davon abgeleitete konkrete Parser
    - String keyword: NodeName des zu parsenden Nodes
    - Childnodes und deren Values auslesen
  - Parserschachtelung
    - FriendListParser nutzt den UserProfileParser
    - PinnboardParser nutzt den ContentParser
    - BlipsParser nutzt den ContentParser
  - Objektgenerierung

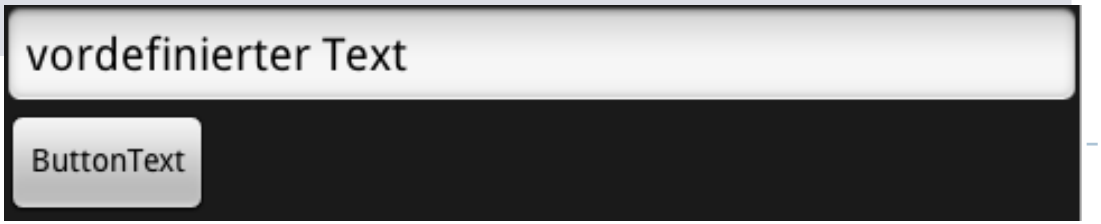
Darstellung der Serverantwort (Views)

- SDK beinhaltet vorgefertigte Komponenten/Widgets
  - Layouts etc.
  - Galleryansicht etc.
- Datenaufbereitung in den Activities und Adaptern
- Komplette View-Gestaltung via XML
  - Teilweise Unterstützung durch Eclipse-Designer
- Ähnliche Style-Definition wie CSS

Beispiel einer View

XML-Code:

```
<?xmlversion="1.0" encoding="utf-8"?>
<LinearLayout
 xmlns:android="http://schemas.android.com/apk/res/android"
 android:layout_width="fill_parent"
 android:layout_height="wrap_content"
 android:orientation="vertical">
  <EditTextandroid:text="vordefinierter Text"
   android:id="@+id/EditText01"
   android:layout_width="fill_parent"
   android:layout_height="wrap_content" />
  <Button style="@style/styleDef" android:text="ButtonText"
   android:id="@+id/Button01"
   android:layout_width="wrap_content"
   android:layout_height="wrap_content" />
</LinearLayout>
```



Ergebnisse

- Verlauf der Entwicklungsphase
- Bewertung Android SDK
- Erreichte Funktionalitäten
- Ausblick

Verlauf der Entwicklungsphase

- Test auf verschiedenen Geräten
  - Android 2.0 (Motorola Milestone)
  - Android 1.6 (HTC G2, HTC Tattoo)
  - Android SDK-Emulator
  - Ausführliche Logging-Informationen
  - Test-Suite beim UMS
- u.a. Context-Broker Positionssimulator

Bewertung Android SDK

- Volle Java-Funktionalität (Vergleich JavaME)
- Android SDK benötigt relativ lange Einarbeitungszeit
- Vordefinierte Komponenten
- Größtenteils Smartphone übergreifend
- Entwicklung noch in den Anfängen, jedoch großes Potenzial
- Gute Basis für ortsbasierte Dienste

Erreichte Funktionalität

- Nicht umgesetzte Funktionen
  - Benutzerverifikation
  - Freundesverwaltung
  - Navigationsfunktion
- Umgesetzte Funktionen
  - Serverkomponenten
  - Radar wird richtig positioniert und mit Blips gefüllt
  - Content anlegen bzw. hochladen
  - Freundesliste, Profile und Content anzeigen

Ausblick

- Modularer Aufbau der Software durch Architektur der Wrapper und Parser
- Weitere ortsbasierte Dienste sind denkbar
  - Trigger
  - „Schatzsuche“
  - Routendefinition
  - Darstellung des Inhalts durch „Augmented Reality“
- Durch verteiltes Serversystem sind Anbindungen weiterer Server zu dem System möglich
- Optimierung von Server/Applikation bringt Performancesteigerung