Hochschule für Angewandte Wissenschaften Hamburg
Hamburg University of Applied Sciences

Bericht INF-M3 PO

Amine El Ayadi

Pervasive Gaming Framework

iLife

Fakultät Technik und Informatik
Department Informations- und
Elektrotechnik

Faculty of Engineering and Computer Science
Department of Information and
Electrical Engineering

Amine El Ayadi

Pervasive Gaming Framework
iLife

Ausarbeitung eingereicht im Rahmen der Veranstaltung Projekt
im Studiengang Master Informatik
am Department Informatik
der Fakultät Technik und Informatik
der Hochschule für Angewandte Wissenschaften Hamburg

Betreuender Prüfer : Prof. Dr. Olaf Zukunft

Abgegeben am 8.März 2009

Inhaltsverzeichnis

1. Einführung und Motivation

1.1. Zielsetzung und Motivation

Derzeit verfügt jeder jüngere Mensch über mindestens ein Handy. Das Nutzen des Handys beschränkt sich nicht mehr nur auf das Telefonieren, vielmehr hat sich der Focus des mobilen Geräts verändert.

Die Funktionalität weitet sich auf Kalender- und Kamera-Funktionalitäten, sowie Internetzugang aus. Zudem bietet das Internet immer mehr interessante Dienste, welche über das Handy anwendbar sind und den Benutzer bereichern.

In diesem Zusammenhang können Handy-Anwendungen entwickelt werden, die das Leben einfacher gestalten bzw. organisieren, indem ausgewählte Dienste auf dem Handy intelligent genutzt werden.

Google zählt zu den besten Dienstanbietern, die nützliche und kostenlose Online-Dienste anbieten (z.B. Google Calendar, Blogger, Picasa, Google Maps, YouTube). Außerdem bietet Google die Handy Plattform Android, die auf dem Linux-Betriebssystem basiert. Ein großer Teil der Software ist frei und Open Source.

Im Rahmen des Projekts wurden das Konzept und die grundlegenden Bausteine der mobilen Anwendung iLife erstellt. iLife soll das Leben des Menschen vereinfachen, indem Google Dienste kombiniert werden. Dabei hat iLife den Vorteil, dass ein mobiles Gerät alle benötigten Informationen für die Google-Dienste enthalten kann (z. B: Fotos, Kalender Einträge, SMS, MMS usw.)

1.2. Gliederung

Die Arbeit gliedert sich in vier Kapitel. Kapitel 1 gibt einen kurzen Überblick über die Motivation, Zielsetzung und den Aufbau der Arbeit. Kapitel 2 stellt die Idee und eine vollständige Beschreibung der intelligenten Anwendung iLife vor. In Kapitel 3 werden die Entwicklungsumgebung sowie die verwendeten Techniken dargelegt, und in Kapitel 4 folgen das Konzept und die Implementierung der grundlegenden Bausteine, die in iLife benötigt werden. Abschließend wird das Projekt in Kapitel 5 zusammengefasst. Es werden die nächsten erforderlichen Schritte aufgezeigt und ein Ausblick gegeben.

2. Definition

Die Anwendung iLife steht als Abkürzung für „intelligent life“ und enthält drei Hauptkomponenten: iCalendar, iBlogger, iOrganizer. Die drei Komponenten arbeiten zusammen und werden anschließend mit entfernten Applikationen (Google Calendar, Blogger, Picasa, Google Maps und/oder YouTube) synchronisiert. Dabei soll der iCalendar alle Termine, Events, SMS, MMS, Fotos und Videos eines Users verwalten (s. Abb. 2.1), sie anzeigen und synchronisieren. Dazu soll der iCalendar die Profile des Handys anhand der eingetragenen Termine aktualisieren. Der iBlogger soll aus dem iCalendar alle Aktivitäten eines Users zusammenstellen und daraus eine Post erzeugen, mit der Möglichkeit, Beschreibungen und Kommentare hinzufügen. Der iOrganizer soll durch Web Semantic Verfahren alle ähnlichen Termine einer Gruppe oder neue übereinstimmende Termine mit anderen Usern finden. Dazu soll iOrganizer dem User die Möglichkeit geben, Events intelligent zu publizieren. Der interessanteste Teil der Anwendung iLife ist der iOrganizer, dabei wird das Handy eine wichtigere Rolle im Leben seines Benutzers spielen, in dem es Denk- und Merkleistungen übernimmt, und somit wird das Benutzer-Leben spannender. Der User bleibt auf dem Laufenden und wird bei jeder Neuigkeit im „Virtuellen“ Netzwerk (Kalender) benachrichtigt, z. B: Zwei User Max und Dirk tragen einen ähnlichen Termin in deren Kalender ein, Max gibt Party als Termin ein, und Dirk gibt Feier ein, dann soll durch die Semantic Web Techniken diese Ähnlichkeit erkannt werden, und alle anderen User aus der gleichen Kalender Gruppe sollen dann benachrichtigt werden. Trägt ein anderer User Hans einen ähnlichen Termin ein, so werden wieder die restlichen User der gleichen Kalendergruppe noch einmal benachrichtigt. Somit hat die Anwendung iLife einen Termin entdeckt und organisiert. Ein weiterer interessanter Teil von iOrganizer ist der Event Publisher: der User stellt ein Event ein, und iLife kümmert übernimmt die Suche nach den Gästen, dabei werden User aus den Social Network nach ähnlichen Interessen mit dem Gastgeber (z. B: Musikgeschmack, Hobbys..) sortiert und anschließend eingeladen.

Abbildung 2.1: iLife Ansicht & Benutzeroberfläche

3. Konzept

3.1 iCalendar

- **iCalendar - Ansicht:**

Der Benutzer erhält eine Übersicht über einen Kalender. Zunächst kann sich der Benutzer zwischen Tages-, Wochen- oder Monats-Ansicht wählen und zu einem beliebigen Tag in der Vergangenheit und Zukunft gehen.

In einem Kalendertag sieht der Benutzer folgende Einträge nach der Uhrzeit sortiert (s. Abb. 2.2):

- an diesem Tag empfangene SMS, die er erhalten hat.

- an diesem Tag empfangene MMS, die er erhalten hat.

- Fotos, die er an diesem Tag gemacht hat

- Videos, die er an diesem Tag aufgenommen hat.

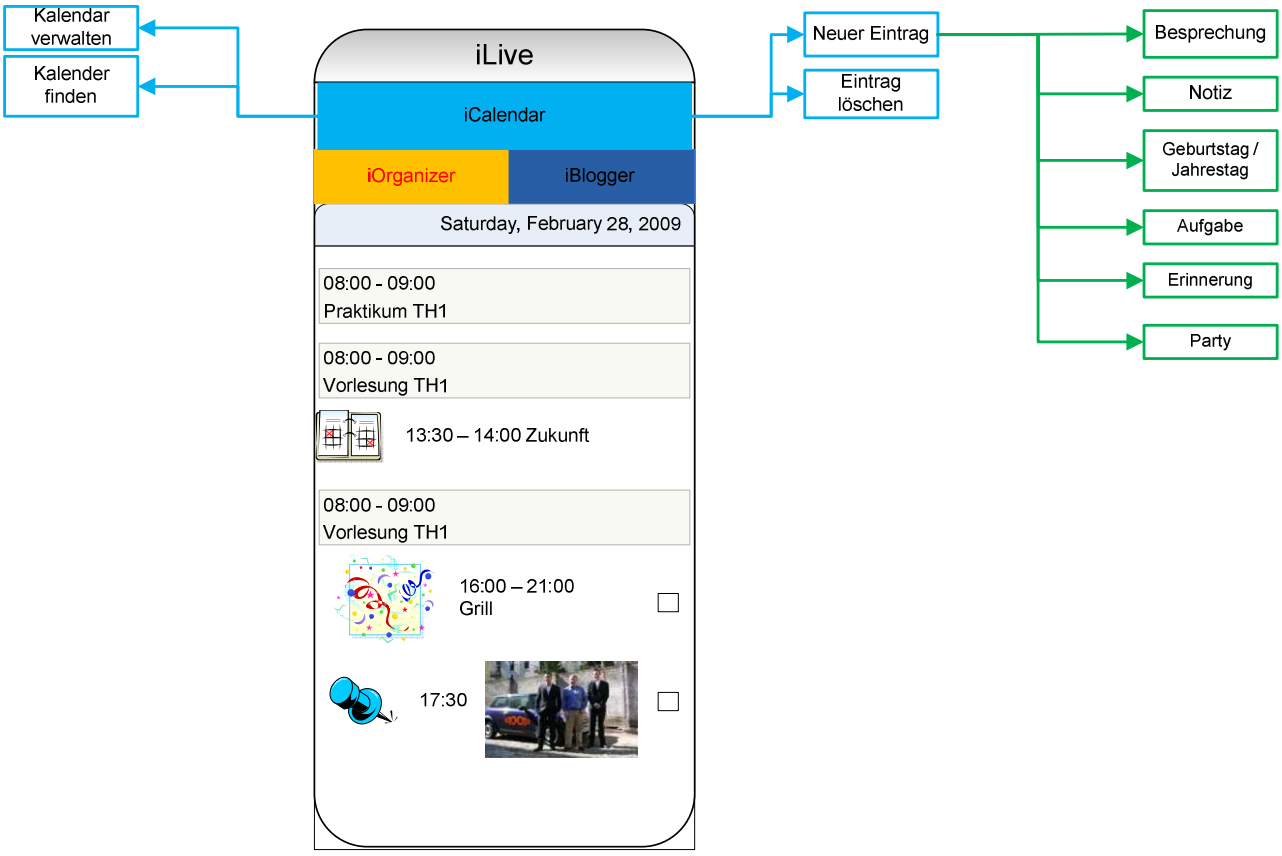

Abbildung 2.2: iLife Tagesansicht

- **iCalendar - Einträge hinzufügen:**

Der Benutzer kann neue Termine (Events) erstellen. Jedes Event ist kategorisiert, und der Benutzer kann für jede Kategorie ein Profil erstellen, welches das Verhalten des Handy währenddessen beschreibt, (z.B. Lautstärke, Klingelton, online/offline usw.) Dabei hat er die Möglichkeit, eins der folgenden Kategorien zu wählen:

- Besprechung: Bei Besprechungen kann der Benutzer einen Betreff, Ort, Datum, Anfangszeit und Ende eingeben. Dabei wechselt das Handy automatisch auf das entsprechende Profil.

- Notiz: Bei Notizen soll der Benutzer einen Betreff, Ort, Datum, Anfangszeit und Ende eingeben.

- Geburtstag / Jahrestag: Hier soll der Benutzer den Anlass und das Datum eingeben.

- Aufgabe: Bei Aufgaben soll der Benutzer den Betreff und das Fälligkeitsdatum eingeben. Zusätzlich kann der Benutzer einen Erinnerungszeitpunkt eingeben.

- Erinnerung: Der Benutzer soll an dieser Stelle einen Betreff wählen und ein Erinnerungsdatum plus Uhrzeit eingeben.

- Party: Bei Parties soll der Benutzer einen Betreff, Ort, Datum, Anfangszeit und geplantes Ende eingeben. Dabei wechselt das Handy automatisch auf das entsprechende Profil.

- **iCalendar - Einträge in Google-Kalender veröffentlichen:**

Der Benutzer kann optional Termine aus dem iCalendar in seinem Google-Kalender veröffentlichen. Hierfür werden zuerst die zu veröffentlichen Termine gewählt, und im Anschluss der Zielkalender bei Google. Der Benutzer habe beispielsweise zwei Google-Kalender, einen für seine Meeting- und Geschäftsplanung und einen für seine Hochschulplanung. Seine Arbeitskollegen und Vorgesetzten hätten Zugriff auf den Meeting- und Geschäftsplanungs-Kalender und seine Kommilitonen auf den Hochschulplanungs-Kalender. Termine wie “mit Kommilitonen etwas trinken gehen“ könnte der Benutzer nur dem Hochschulplanungskalender hinzufügen, damit seine Arbeitskollegen und Vorgesetzten keine Einsicht in diesen Termin haben. Weitere Termine, wie z.B. Master-Kolloquium, könnten dagegen in beiden Kalendern veröffentlicht werden.

- **iCalendar - Synchronisieren:**

Der Google Kalender wird mit dem Kalender des mobilen Geräts nach jedem Starten der iCalendar Anwendung synchronisiert. Zusätzlich hat auch der Benutzer die Möglichkeit, über das User Interface die Synchronisation zu starten.

3.2 iBlogger

- **iBlogger - Einträge in einem Blog veröffentlichen:**

Der Benutzer hat die Möglichkeit, die Einträge aus dem iCalendar in einem Blog zu veröffentlichen. Der Benutzer wählt aus dem iCalendar Termine, Fotos, Videos, SMS und/oder MMS aus, die er publizieren möchte und kann zusätzlich Kommentare und Beschreibungen hinzufügen (s. Abb. 2.3).

Da ein Blog ein Tagebuch darstellt, ist es nur erlaubt, Einträge aus der Vergangenheit im Blog zu publizieren. Diese iCalendar Einträge werden gespeichert und anschließend im Google Blogger veröffentlicht. Zudem kann zu jedem Eintrag ein Kommentar vom Benutzer verfasst werden, der im Blog ebenfalls zu sehen ist. Alle im Blog veröffentlichen Fotos werden mit der Ortsbestimmung dargestellt, sodass jeder sehen kann, wo genau die Fotos entstanden sind. Dabei erhält der Benutzer einen Link zu Google Maps, um den Ort auf der Karte anzeigen zu lassen. Da die Blogger API die Funktion „Foto hochladen“ nicht unterstützt, werden die Fotos in Picasa hochgeladen und anschließend in dem Blogger verlinkt. Ebenfalls können Videos über YouTube hochgeladen und ebenfalls verlinkt werden.

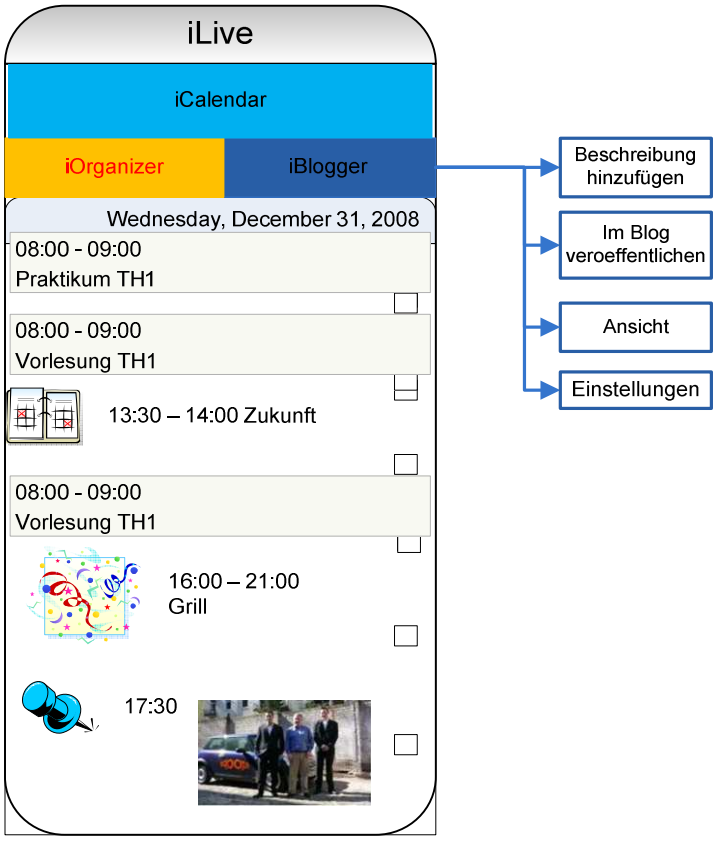

Abbildung 2.3: Blog Funktionalitäten- Ansicht

- **iBlogger - Ortsbestimmung bei Aufnahme von Fotos/ Videos:**

Bei Aufnahme von Fotos oder Videos muss das mobile Gerät seinen Aufenthaltsort bestimmen. Dieser wird mit dem Foto oder Video abgespeichert.

3.3 iOrganizer

- **iOrganizer - Ähnliche Termine von Gruppen finden:**

Der Benutzer soll die Möglichkeit haben, ähnliche Termine innerhalb von vordefinierten Gruppen oder selektierten Benutzern zu erkennen. Es wird nach ähnlichen Terminen gesucht, die am selben Tag und in ähnlichen Zeiträumen stattfinden (s. Abb. 2.4). Hat z.B. ein Benutzer aus der Gruppe Fußballverein einen Termin am 11.12.2008 im Zeitraum 22:00 – 24:00 Uhr mit der Terminbezeichnung „Kinoabend“ und ein weiterer Benutzer derselben Gruppe einen Termin am gleichen Tag mit dem Zeitraum 20:00 – 22:00 Uhr und der Bezeichnung „Ins Kino gehen“, dann soll der Benutzer sehen können, dass diese beiden Benutzer aus derselben Gruppe zwei sehr ähnliche Termine am gleichen Tag haben. Jetzt kann der Benutzer sich mit den beiden anderen zu einem Kinobesuch verabreden.

- **iOrganizer - Nach Begriffen in anderen Google-Kalendern suchen:**

Der Benutzer soll nach Suchbegriffen in Google-Kalendern von anderen Benutzern suchen können. Hierzu kann der Benutzer bestimmen, ob er bei nur einem anderen Benutzer nach dem Suchbegriff sucht oder in einer Gruppe. Beide Optionen sind interessant, wenn der Benutzer z.B. sehen möchte, wer von seinen Kommilitonen TT2 lernt, dann sucht er in der gesamten Gruppe. Falls der Benutzer aber nur wissen möchte, wann eine bestimmte Person wieder kocht, dann ist es sinnvoll nur bei dem einen Benutzer zu suchen.

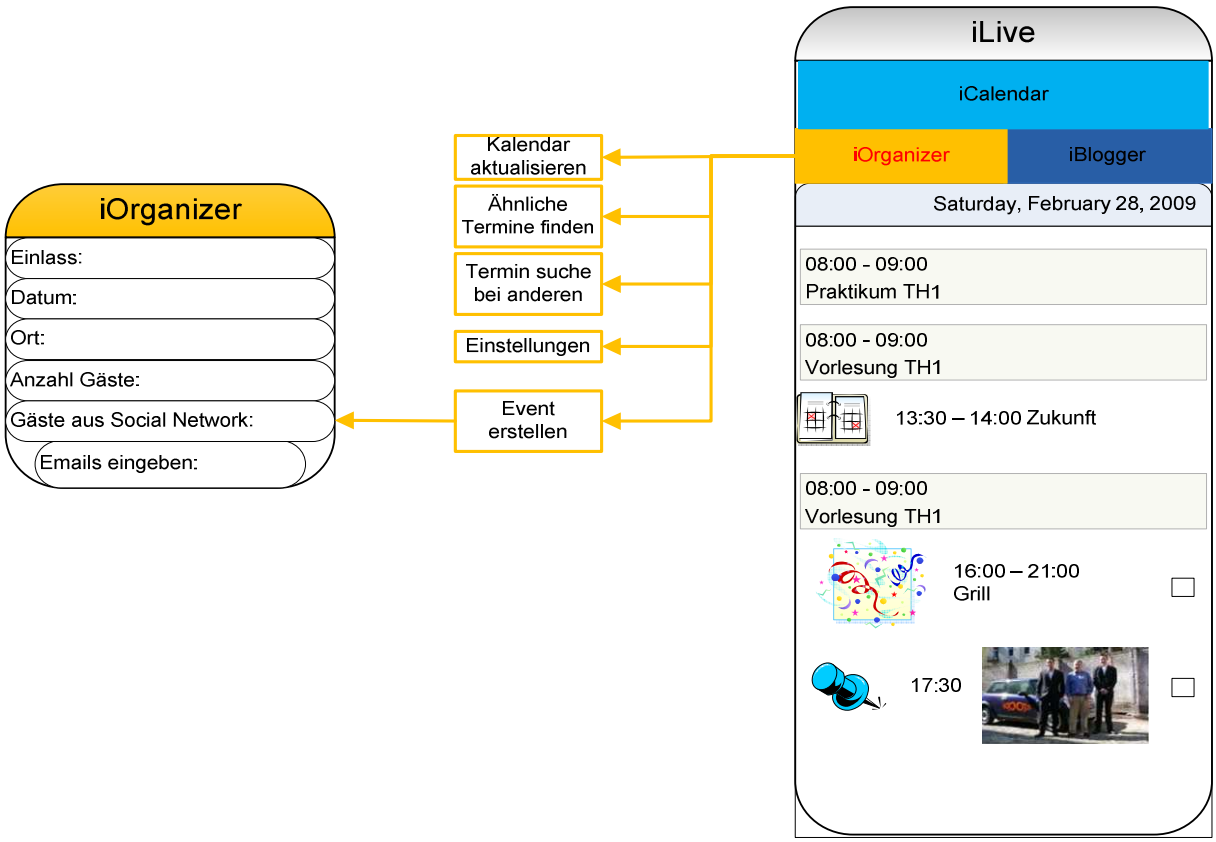

Abbildung 2.4: iLife-iOrganizer

- **iOrganizer - Neue übereinstimmende Termine anzeigen:**

Nach jeder Kalendersynchronisation werden dem Benutzer alle ähnlichen Termine angezeigt, die einer seiner Kontakte mit dem Benutzer gemeinsam hat.

- **iOrganizer - Termine von anderen Benutzern ansehen:**

Der Benutzer soll die Möglichkeit haben Terminkalender von einem anderen Benutzer einzusehen, damit der Benutzer beispielsweise sehen kann, wann sein Chef wieder geschäftlich verreisen muss.

- **iOrganizer - Event Publisher:**

Der Event-Publisher bietet dem Benutzer die Möglichkeit, alle seine Events für eine bestimmte Gruppe zu publizieren. Bei dieser Gruppe kann es sich um eine vordefinierte Gruppe oder Benutzer aus der Freundesliste seines Kalenders handeln. Ebenfalls können für die Freunde des Freundes die Events veröffentlicht werden. Es kann bestimmt werden, auf welcher Netzwerk-Ebene die Events veröffentlich werden sollen, wobei nur Freunde mit ähnlichen Interessen erkannt und eingeladen werden. Der Benutzer kann ebenfalls alle Kontakte aus seinem Email-Account über seine Events informieren.

4. Umsetzung

In diesem Abschnitt werden zuerst die Plattform und die Entwicklungs-Umgebung definiert bzw. vorgestellt, danach wird die Architektur beschrieben. Anschließend wird der Einsatz der Proxy-Server erklärt. Zuletzt werden die implementierten Basisanwendungen beschrieben. Dabei werden alternative Lösungsmöglichkeiten benannt und der Grund dargelegt, aus dem sich das iLife Team für diese Lösungen entschieden hat.

4.1 Plattform und Entwicklungs-Umgebung

4.1.1 Android

Android ist eine vollständige Plattform für mobile Geräte. Sie bringt ein Betriebssystem, eine Laufzeitumgebung, ein Anwendungs-Framework und fertige Kernanwendungen mit. Ist passende Hardware vorhanden, reicht es also Android zu installieren, um ein voll funktionsfähiges Handy zu erhalten. Es kann in Java programmiert werden. Alle Möglichkeiten der Hardware können genutzt, die meisten Applikationen nach Belieben ergänzt oder ersetzt werden (auch Telefonieapplikationen). Zum einen erhält der Entwickler mit Android Freiheiten, die er auf anderen Plattformen nicht hat, und zum anderen erhält er Werkzeuge, mit denen er mobile Anwendungen komfortabel entwickeln kann.

4.1.2 Entwicklung

Anwendungen in Android können mit der SDK entwickelt und mit dem mitgelieferten Emulator getestet werden.

Android Runtime: Alle Android-Anwendungen laufen in einem eigenen Prozess in einer Instanz der Android-Laufzeitumgebung. Die Laufzeitumgebung heißt Dalvik Virtual Machine (VM). Hierbei handelt es sich um eine von Google entwickelte, für mobile Geräte optimierte virtuelle Maschine, die den Android eigenen Bytecode ausführt, den so genannten Dalvik-Bytecode. Wichtig: Die Dalvik VM bringt nicht alle Java APIs einer Java Runtime mit. Die wichtigsten APIs sind aber vorhanden, und man kann eigene JAR-Bibliotheken mit seiner Anwendung mitliefern.

Anwendungs-Framework: Das Anwendung- Framework enthält die grundlegenden Bausteine und APIs, mit denen Android-Anwendungen entwickelt werden. Es ist damit die wichtigste Schnittstelle für alle Android Anwendungen.

Anwendungen: Die Android-Plattform beinhaltet bereits die wichtigsten Anwendungen für ein Handy, wie z. B: Telefonie, Kontakte, Kalender, Webbrowser, Google Maps und weitere. Alle Anwendungen werden in Java geschrieben und verwenden die Android Runtime. Ein klarer Vorteil von Android ist, dass Anwendungen nicht voneinander isoliert sind, sondern über einfache Mechanismen lose miteinander gekoppelt werden können. Entwickler können, wenn die Anwendungen entsprechend gebaut sind, einzelne Funktionalitäten anderer Anwendungen

für ihre Anwendung nutzen. Beispiel: Ich möchte in meiner Anwendung die Möglichkeit bieten, einen Kontakt auszuwählen. Anstatt dafür meinen eigenen Dialog zu kreieren, gehe ich einfach in die Kontaktverwaltungsanwendung. Ich wähle dort den Kontakt aus, und die aufgerufene Anwendung liefert mir den ausgewählten Kontakt zurück. Auf diese Weise kann ich andere Anwendungen als Dienste betrachten und bei Bedarf in meine Anwendung integrieren [Java Magazin].

4.2 Architektur

In diesem Abschnitt wird die iLife-Architektur vorgestellt (s. Abb. 4.1). Für eine Beschreibung des Zusammenspiels der einzelnen Komponenten wird auf Kapitel 3 verwiesen.

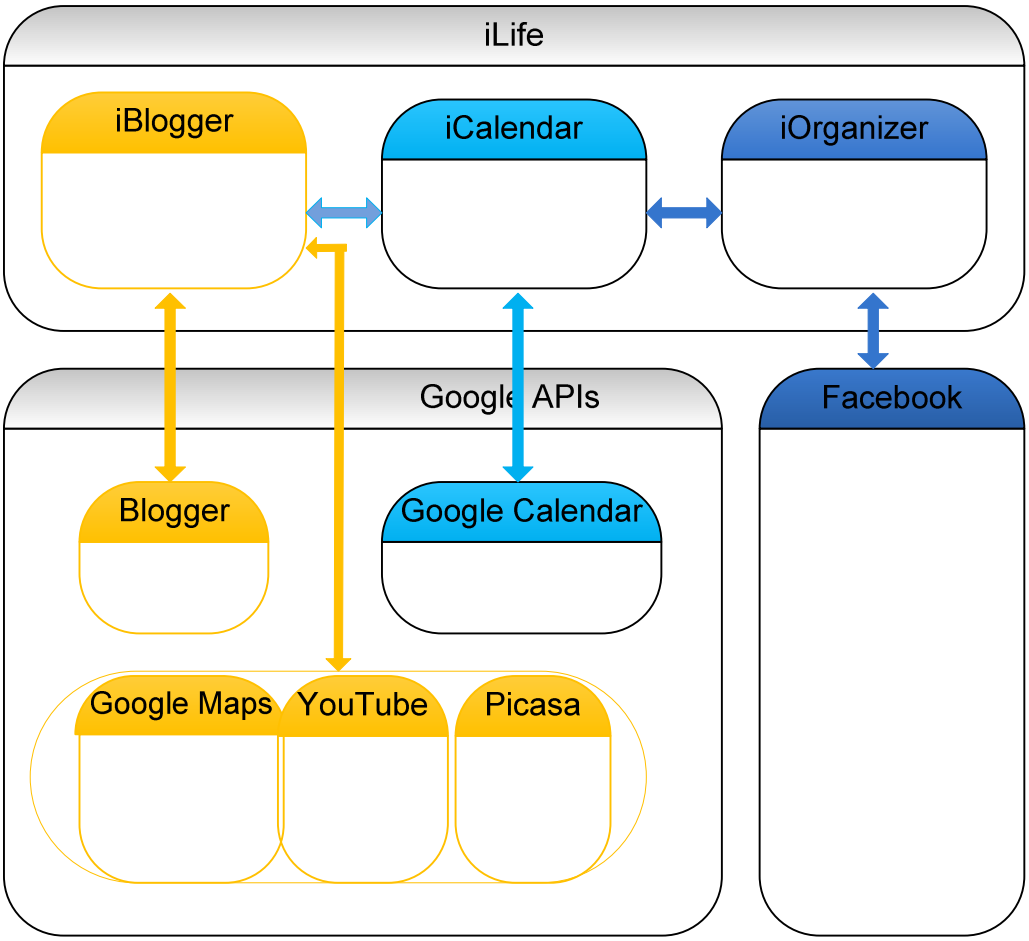

Abbildung 4.1 iLife-Architektur

4.3 Proxy-Server

Eine der Basisapplikationen für iLife ist die Kalender-Anwendung iCalendar. Diese Anwendung sollte im Rahmen dieses Projekts entwickelt werden, weil der Standard Android-Kalender keine Open Source Anwendung ist.

Der iCalendar soll mit dem Google-Kalender verbunden und synchronisiert werden, aber aufgrund der Unkompatibilität der Google API's mit dem Plattform (da die Dalvik VM nicht alle Java APIs einer Java Runtime anbietet) sollte ein Proxy-Server entwickelt werden (s. Abb. 4.2). Dieser Server wird später verwendet, um den Umstand der Suche nach ähnlichen Terminen bzw. Interessen auf das Handy zu vermeiden.

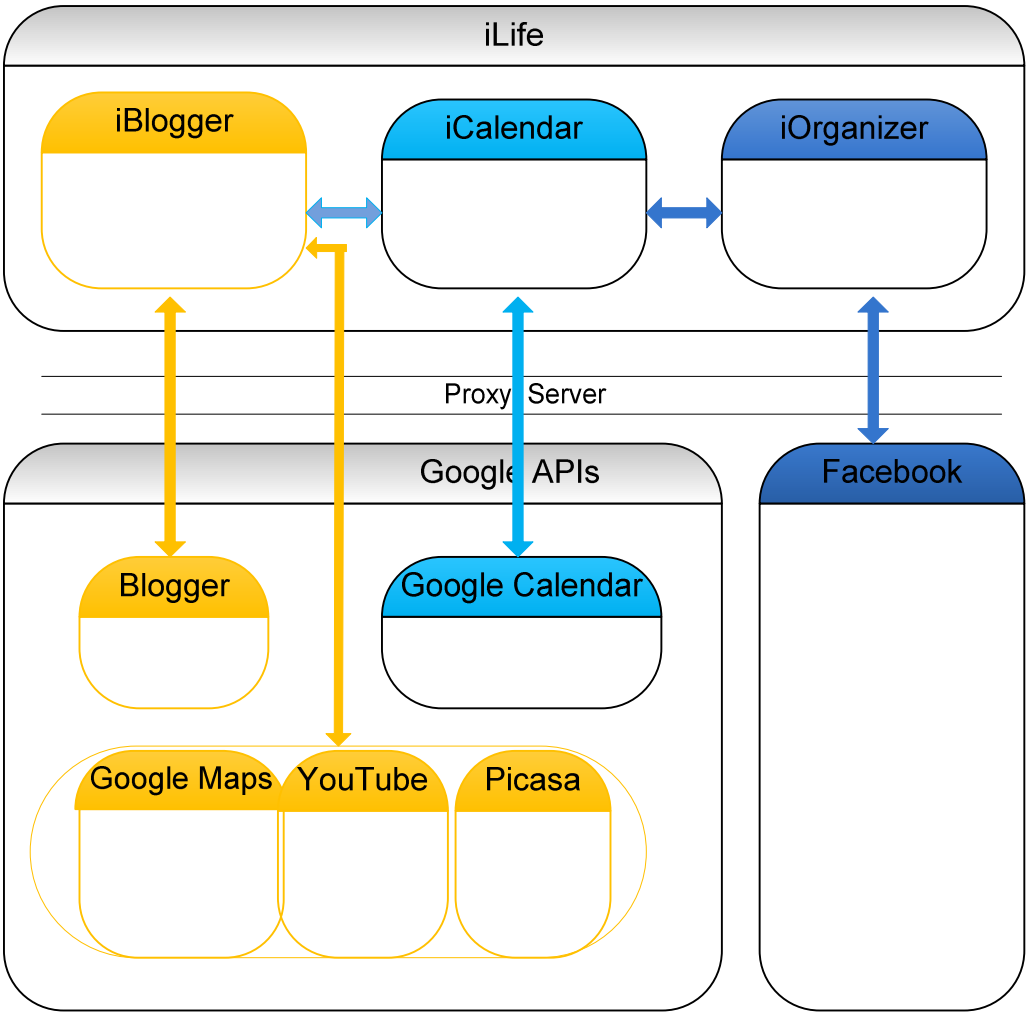

Abbildung 4.2 Proxy-Server-Architektur

Der iLife-Proxy-Server besteht aus zwei Komponenten, dem iCalender-Proxy-Server und dem iOrganizer-Proxy-Server, wobei der iCalender-Proxy-Server im Rahmen dieses Projekts implementiert wurde.

Der iCalender-Proxy-Server implementiert die iCalender Anwendung auf Basis von Apache

Tomcat 6 und Apache Axis2 (Apache eXtensible Interaction System). Dabei werden folgende

Methoden angeboten:

- Termine ermitteln (getEvent):

Diese Methode erwartet ein Datum, UserId und Passwort als Eingabeparameter und gibt für den User die Termine zurück, die er an dem Tag eingetragen hat:

```
public CalendarEventFeed getEvent(String date,String userid,String password){
// Set up the URL and the object that will handle the connection:
URL feedUrl = new
URL("http://www.Google.com/calendar/feeds/default/private/full");

CalendarQuery myQuery = new CalendarQuery(feedUrl);
myQuery.setMinimumStartTime(DateTime.parseDateTime(date+"T00:00:00"));
myQuery.setMaximumStartTime(DateTime.parseDateTime(date+"T23:59:59"));

CalendarService myService = new CalendarService("exampleCo-exampleApp-1");
myService.setUserCredentials(userId, password);

// Send the request and receive the response:
CalendarEventFeed resultFeed = myService.query(myQuery, Feed.class);
Return resultFeed;}
```

- Termine erstellen (createEvent):

Diese Methode erwartet Anfangs-, Endzeit, Titel, Content, UserId und Password als Eingabeparameter und gibt das erstellte Terminobjekt zurück:

```
public CalendarEventEntry createEvent(String startDate, String endDate,String title, String content,String userid,String password){
// Set up the URL and the object that will handle the connection:
URL postUrl =
  new URL(http://www.Google.com/calendar/feeds/+UserId+"/private/full");
CalendarEventEntry myEntry = new CalendarEventEntry();

myEntry.setTitle(new PlainTextConstruct("Termin Titel"));
myEntry.setContent(new PlainTextConstruct("Beschreibung"));

DateTime startTime = DateTime.parseDateTime(startDate);
DateTime endTime = DateTime.parseDateTime(endDate);
When eventTimes = new When();
eventTimes.setStartTime(startTime);
eventTimes.setEndTime(endTime);
myEntry.addTime(eventTimes);

// Send the request and receive the response:
CalendarEventEntry insertedEntry = myService.insert(postUrl,
myEntry);
Return insertedEntry;}
```

Der iOrganizer-Proxy-Server soll später folgende Methoden anbieten:

- Ähnliche Termine ermitteln:

Diese Methode erwartet einen Iterator mit allen Terminen und gibt einen Iterator mit ähnlichen Terminen zurück.

- Profile mit ähnlichen Interessen definieren:

Diese Methode erwartet einen Iterator mit allen Profilen der Freunde auf einem Netzwerk und gibt einen Profilen-Iterator mit den Profilen, die ähnliche Interessen haben, zurück.

4.4 iCalendar

Es war geplant, den iCalender auf dem Android Kalender basieren zu lassen. Genauere Recherchen haben jedoch ergeben, dass der Android Kalender nicht Open Source ist und sich von daher nicht eignet.

Nachdem auch die Suche nach einer geeigneten Software, welche als Grundlage für den iCalender dienen kann, erfolglos blieb, standen noch folgende Alternativen zur Auswahl:

- Eine Kalenderanwendung zu entwickeln, wobei die Daten lokal auf dem Gerät gespeichert werden und dann mit dem Google Kalender Synchronisiert werden:
  - Vorteil: Die Daten werden an zwei Stellen gesichert, lokal und entfernt.
  - Nachteil: Die Synchronisation ist aufwändig zu implementieren und bietet keine Vorteile für die Anwendung iLife gegenüber der Alternativlösung.
- Eine Kalenderanwendung zu entwickeln, die als client- Anwendung und direkt auf den Google Kalender zugreift, ohne die Daten lokal zu speichern.
  - Vorteil: Die Implementierung dieser Lösung ist weniger aufwändig.
  - Nachteil: Es können Termine nur eingetragen werden, wenn der User Online ist, aber das ist kein großer Nachteil für die iLife, denn eine Internetverbindung ist eine Grundvoraussetzung für iLife.

Die iLife Team Mitglieder haben sich dann entschlossen, die zweite Lösungsvariante zu verwenden. Die Implementierung war nicht problemlos, weil festgestellt wurde, dass die Google API’s unkompatibel mit der Android-Plattform sind (da die Dalvik VM nicht alle Java APIs einer Java Runtime anbietet). Da aber in der Konzeptionsphase eine Client-Server-Architektur geplant wurde, um den Umstand der Suche nach ähnlichen Terminen bzw. Interessen auf dem Handy zu vermeiden, wurde diese Architektur erweitert, um das Unkompatibilitäts-Problem zu lösen. Dabei sollte der Server zwei zusätzliche Methoden anbieten:

- createEvent (siehe Abschnitt 4.3.2 Proxy-Server)
- getEvent (siehe Abschnitt 4.3.2 Proxy-Server)

Die zwei Methoden wurden auf der Client-Seite aufgerufen. Dadurch wurde eine der Basisanwendungen für iLife entwickelt.

4.5 Social Network

Die Anbindung an das Social Network ist ebenfalls wichtig in iLife, denn der Event-Publisher von iOrganizer benötigt mehrere, unterschiedliche Informationen über die Gäste, die meistens nur von einem Social Network angeboten werden können. Zum Beispiel wird bei der Suche nach passenden Gästen versucht, Gemeinsamkeiten zwischen diesen Personen zu finden. Die Gemeinsamkeiten können sich auf verschiedene Beziehungskategorien beziehen, wie Freunde, Arbeitskollegen, gleiche Interessensgebiete, Visionen, Geschäftsbeziehungen, dieselbe Universität, Konferenzen, etc.

Facebook ist eines der erfolgreichsten Social Networks und enthält alle nötigen Informationen für den iOrganizer. Außerdem bietet Facebook mehrere APIs, um den Zugriff auf die nötigen Informationen zu ermöglichen:

fb4j Facebook Client for Java (http://fb4j.sourceforge.net/)

fb4j ist eine Facebook-API für Java-Entwickler. fb4j bietet einen einfachen Zugriff auf alle Funktionen der Facebook-API, wobei es einfach zu bedienen ist und volle Unterstützung für die Authentifizierung bietet.

JavaBook (http://code.Google.com/p/javabook/)

JavaBook ist eine neue Java Bibliothek für Facebook. Die aktuelle Version ist ein Alpha, dadurch ist es nicht auszuschließen, dass die Funktionalitäten dieser Bibliothek fehlerhaft sind. Für Entwickler steht der Source-code dieser Bibliothek auf den Google-Code zur Erweiterung frei.

Facebook Java API Community Project (http://code.Google.com/p/facebook-java-api/)

Facebook Java API ist ein Open-Source-Projekt, das die neueste offizielle Java-Clientversion als Ausgangspunkt nutzt. Es enthält zusätzliche Funktionen (wie z. B. integrierte Unterstützung für "feed.publishTemplatizedAction").

Das iLife Team hat sich für die erste API (fb4j Facebook Client) entschieden, denn diese bietet eine volle Unterstützung für die Authentifizierung für Web und Desktop Anwendungen an. Die Funktionalität des fb4j Facebook Clients wurde getestet, in dem einige Methoden auf der Client-Seite implementiert wurden (z. B: getAllFriends, getProfileById(String id)…).

Die Anbindung dieser Methoden mit Hilfe des iOrganizer kann in einer späteren Projekt-Phase stattfinden, denn der iOrganizer wurde noch nicht implementiert.

5. Schluss

5.1 Fazit

Die Arbeit beschreibt das Konzept der mobilen Anwendung iLife und die Implementierung der Basisapplikationen, die für die Realisierung von iLife notwendig sind. Dabei wurde eine Kalender-Anwendung für Android realisiert und die Schnittstellen zum Google-Kalender und zum Social Network „Facebook" implementiert.

Die Implementierung der Kalender-Anwendung war notwendig, da der Standard Android-Kalender keine Open Source Anwendung ist. Der Proxy-Server war hilfreich, um das Problem der Unkompatibilität von den Google API's mit dem Android Plattform zu lösen. Zudem wurde die API-Anbindung des Bloggers und von Facebook exemplarisch durchgeführt. Unter anderem wurden erste Versuche, Fotos über Picasa, Ortsinformationen über Google Maps, Videos über YouTube und Informationen über Freunde aus dem Social Network „Facebook" zu ermitteln, erfolgreich realisiert (http://aelayadi.blogspot.com/).

5.2 Weiterentwicklung

Nachdem die Basis Applikationen für iLife teilweise implementiert wurden, stellt sich iLife als ein ideales Thema für die Masterarbeit dar. Als erster Schritt soll die Anzeige im iCalender mit Bildern, Videos, SMS usw. erweitert werden, daran kann anschließend der iBlogger implementiert werden.

Bei der Implementierung des iBlogger ist die Anbindung von drei APIs notwendig:

- Picasa: Die Handy Fotos werden in Picasa hochgeladen, damit sie später im Google Blogger verlinkt werden können. Die Picasa API wird verwendet, weil die Google Blogger API die Funktion „Foto hochladen" nicht unterstützt.

- YouTube: Die Videos werden in YouTube nach dem gleichen Prinzip wie die Fotos in Picasa behandelt, damit sie im Google Blogger verlinkt werden können. Die YouTube API wird verwendet, weil die Google Blogger API die Funktion „Videos hochladen„ nicht unterstützt.

- Google Maps: Die Ortsinformationen von den aufgenommenen Fotos und Videos werden an Google Maps API übermittelt, um den Ort auf dem Google Blogger zu verlinken.

Die Implementierung der iOrganizer erfolgt als zweiter Schritt. Davor ist eine Recherche im Semantic Web Bereich notwendig. Dabei sollten die passenden Text Mining Verfahren, die die Suche nach ähnlichen Terminen bzw. Interessen ermöglichen, definiert werden. Anschließend können diese Verfahren implementiert werden.

Quellen

- **[Java Magazin] Java Magazin, Ausgabe 08/2008**

- **Facebook Java API Community Project:**
  http://code.Google.com/p/facebook-java-api/

- **JavaBook**
  http://code.Google.com/p/javabook/

- **fb4j Facebook Client for Java**
  http://fb4j.sourceforge.net/

- **Google Data APIs**
  http://code.google.com/intl/de-DE/apis/gdata/

- **Differentiating data- and text-mining terminology**
  http://portal.acm.org/