Informationsanbieterzentrierte Spezifikation und Generierung von Informationssystem-Apps

Jonas Pencke, David Wiesner, Hagen Höpfner und Maximilian Schirmer
Bauhaus-Universität Weimar
Bauhausstraße 11
99423 Weimar, Germany
VORNAME.NACHNAME@uni-weimar.de

Schlüsselworte

Mobile Informationssysteme, Nutzergetriebene Programmierung, App-Erzeugung

Zusammenfassung

Mobilgeräte wie z.B. Smartphones werden heutzutage nicht mehr ausschließlich zur Sprachkommunkation eingesetzt. Sie ermöglichen es, zeitnah Informationen an mobile Nutzer zu übertragen. Hierbei spielt der Aufenthaltsort der Nutzer weitestgehend keine Rolle, er/sie ist quasi jederzeit und allerorts erreichbar. Im Gegensatz zu der Einfachheit der Informationskonsumtion ist das Entwickeln von Apps nicht trivial. Hierzu ist Expertenwissen notwendig. Zahlreiche potentielle Informationsanbieter verfügen nicht über die notwendigen Kenntnisse, wenngleich ihre Informationen für zahlreiche Konsumenten interessant wären und es starke strukturelle Ähnlichkeiten zwischen mobil verfügbar gemachten Informationen gibt. Ein weiteres Problem ist, dass unterschiedliche Smartphone-Hersteller dediziert unterschiedliche Programmiersprachen benutzen. In diesem Papier präsentieren wir unseren Ansatz zur anbieterzentrierten Generierung von Apps, wobei ein Hauptaugenmerk auf der Unterstützung heterogener Zielplattformen liegt. Somit ermöglichen wir es technisch nicht versierten Informationsanbietern, ihre eigenen Apps zu erzeugen.

1. EINLEITUNG UND MOTIVATION

Der Einsatz moderner mobiler Endgeräte wie z.B. Smartphones ist nicht mehr auf reine Sprachkommunikation begrenzt. Vielmehr werden zahlreiche Bereiche der Informations- und Kommunikationstechnologien, welche noch vor wenigen Jahren Desktop-Computern vorbehalten waren, unterstützt. Leistungsfähige Geräte wie Apples iPhone oder HTCs Desire ermöglichen den Einsatz von Sofortnachrichtendienste (engl. instant messaging) und klassischen Informationssystemen sowie die Partizipation in sozialen Netzen (Facebook, MySpace, StudiVZ, etc.) somit beinahe jederzeit und allerorts. Im August 2010 veröffentliche Gartner eine Studie [11], nach der 19% aller neu verkauften Mobiltelefone derartige Smartphones sind. Neben der Tatsache, dass Smartphones den mobilen Informationszugriff ermöglichen, bieten sie auch die Möglichkeit der sogenannte Push-Notifikation. Dabei werden Neuigkeiten, Änderungen oder Hinweise direkt vom Anbieter auf das Gerät transferiert, wodurch der Benutzer nicht explizit und regelmäßig danach suchen muss. Von diesem weiteren Informationsverteilungprozess profitieren auch die Informationsanbieter, da sie somit ihre Informationskonsumenten nahezu jederzeit und direkt erreichen können.

Im Gegensatz zur Entwicklung von Softwareprodukten für Desktop-Computer ist das Programmieren von Anwendungen für Smartphones (den sogenannten Apps) an die Verwendung einer dedizierten Programmiersprache gebunden. iOS-Entwickler verwenden Objective-C. Apps für Smartphones mit Google Android werden in Java implementiert. Geräte mit Microsoft's Windows Phone 7 unterstützen C# und Apps für HPs webOS werden in HTML 5, CSS und JavaScript realisiert. Folglich bedingt der Wunsch nach einer breiten Unterstützung verschiedener Smartphones die Notwendigkeit, dass Entwickler mit mehreren dieser Programmiersprachen vertraut sind. Natürlich eröffnet dies einen neuen Markt für Programmierer, es hält jedoch zahlreiche technisch nicht versierte Informationsanbieter davon ab, ihre Information über das Medium Smartphone anzubieten. Unseres Wissens nach existieren momentan keine Lösungen, welche, vergleichbar zu den im WWW etablierten Redaktionssystemen (engl. content management systems), einen einfachen Zugang für Nicht-Programmierer bieten. Des Weiteren sind Web-basierte Ansätze, welche die Verwendung der Internet-Browser der Smartphones voraussetzen, keine Lösung des Dilemmas. Einerseits bieten diese keine Push-Notifikation, andererseits erhöht das manuelle, wiederholte (meist vergebliche) Suchen nach aktuellen Informationen, die Anzahl von energieintensiven drahtlosen Datenübertragungen.

In diesem Papier präsentieren wir unseren Ansatz, die genannten Probleme durch ein Framework zur Erzeugung von Informationssystem-Apps zu beheben. Durch die Analyse verschiedener existierender Apps haben wir herausgefunden, dass die angebotenen Informationen klassifiziert werden können. Basierend auf dieser Klassifikation haben wir eine modulare Struktur entwickelt, die als Code-Skelett für verschiedene Plattformen realisiert wurde. Mithilfe dieser Code-Skelette kann dann eine App wie folgt erzeugt werden: Zuerst wird die App mithilfe eines domänen-spezifischen, Web-basierten Konfigurationswerkzeugs parametrisiert. Die dabei spezifizierten statischen Informationen werden dem Quelltext hinzugefügt. Dynamische Informationen werden auf der Webseite des Anbieters hinterlegt, wo sie auch gewartet werden können. Anschließend wird die App automatisch kompiliert und zum Download angeboten. Neben dem Vorteil, dass auf diese Weise technisch nicht versierte Anwender wie Musiker, Politiker, etc. ohne Kenntnis einer Programmiersprache Apps erstellen können, vermeidet

GvD Workshop'11, 31.05.-03.06.2011, Obergurgl, Tirol, Austria.

unser Verfahren die potentielle Verletzung von Urheber- und Verwertungsrechten. Alle Informationen, die durch die App angeboten werden, werden auch direkt vom Anbieter bereitgestellt und befinden sich unter dessen Kontrolle[1].

Der Rest des Papieres ist wie folgt strukturiert: Abschnitt 2 beschreibt verwandte Arbeiten. Abschnitt 3 analysiert die Eigenschaften der Informationen, welche üblicherweise durch Informationssystem-Apps angeboten werden. Abschnitt 4 beinhaltet eine kurzen Einblick in unseren Evaluationsprototyp und präsentiert die Architektur unseres App-Erzeugungssystems. Abschnitt 5 fasst das Papier zusammen und gibt einen Ausblick auf Folgearbeiten.

2. VERWANDTE ARBEITEN

Die in diesem Beitrag vorgestellten Forschungsergebnisse können den Forschungsgebieten Code-Generierung (engl. code generation), mobile Informationssysteme und Mensch-Maschine-Interaktion (engl. human-computer interaction) zugeordnet werden.

Code-Generierung wird oft als Teilgebiet der modellgetriebenen Softwareentwicklung [5] angesehen. Hierbei werden domänen-spezifische Sprachen verwendet, um abstrakte Modelle von Software-Systemen zu erzeugen. Aus diesen Modellen erzeugen dann Code-Generatoren den Quelltext, entweder in Teilen oder komplett. Unser Ansatz ist es jedoch, Endbenutzern ohne technisches Wissen die Möglichkeit zu geben, Anwendungen zu erzeugen. Von diesen Benutzern kann nicht verlangt werden, formale Spezifikationen zu verstehen oder zu verwenden. Zudem unterscheiden sich die verfügbaren Smartphones verschiedener Hersteller sehr in den unterstützten Programmiersprachen, wie bereits in Abschnitt 1 erläutert. Für unseren Ansatz hätte dies bedeutet eine sehr große Anzahl von verschiedenen, formal spezifizierten Code-Generatoren und domänen-spezifischen Sprachen zu erzeugen, um eine breite Masse von Smartphones unterstützen zu können. Daher haben wir uns dazu entschlossen, ein gerätespezifisches Code-Skelett anzubieten, gemäß dem Paradigma der generativen Programmierung [2]. Unser „Code-Generator“ parametrisiert und ergänzt das Code-Skelett, welches schließlich kompiliert wird. In der Zukunft werden wir an einer formaleren Spezifikation arbeiten.

Mobile Informationssysteme machen es möglich, Informationen auf mobilen Endgeräten anzubieten. Die größte Herausforderung besteht dabei in der Reduktion des übertragenen Datenvolumens zum und vom Gerät. Die drahtlose Datenübertragung ist unverzichtbar, aber auch sehr energieintensiv [3, 1], vergleichsweise langsam [13] und (je nach Mobilfunkvertrag) teuer. Zudem mindern physikalische Effekte die Verfügbarkeit von drahtlosen Netzen [9]. Es gibt bereits verschiedene Ansätze, um das anfallende Datenvolumen zu reduzieren. Die Forschungsergebnisse reichen dabei von Caching [8, 12] über Hoarding [7] bis hin zur Replikation [4]. In unserem System begegnen wir dieser Herausforderung, indem so viele Informationen wie möglich bereits in der App enthalten sind. Nur dynamische Informationen (s. Abschnitt 3) werden an das mobile Endgerät übertragen, wo sie zudem für den späteren Zugriff zwischengespeichert werden.

Smartphones bieten meist nur sehr kleine Bildschirme und im Vergleich zu normalen Desktop-Computern grundverschiedene Möglichkeiten der Benutzerinteraktion. Entsprechend gilt es, bei Aspekten der Mensch-Maschine-Interaktion auf diese Unterschiede einzugehen [6]. Die Software-Entwicklungsumgebungen der verschiedenen Smartphone-Plattformen bieten bereits vereinheitlichte Interaktionsschemata, an die die Benutzer der jeweiligen Geräte und Plattformen gewöhnt sind. Generell bevorzugen sie eine konsistente Benutzeroberfläche über alle verwendeten Apps [10]. Deshalb haben wir uns dazu entschlossen, bestehende Benuteroberflächen-Toolkits zu verwenden.

[1]Wir nehmen an, dass der Anbieter die entsprechenden Rechte besitzt bzw. entsprechende Verträge hierfür selbst abgeschlossen hat.

3. INFORMATIONEN IN APPS

Durch die Analyse verschiedener Informationssystem-Apps haben wir zwei Klassen von präsentierten Informationen ableiten können:

Statische Informationen sind Informationen, die explizit in den Binärcode von Apps hinein kompiliert werden. Nahezu alle Apps enthalten statische Informationen wie den Namen der App, statisch verknüpfte Bilder, Informationsdialog, usw. Aktualisierungen dieser Informationen sind nur durch Neukompilierung der App möglich. Dafür ist es möglich, statische Informationen auch ohne Netzverbindung zu verwenden.

Dynamische Informationen sind Informationen, die zur Laufzeit heruntergeladen werden. Dies sind beispielsweise Nachrichten-Streams, ortsabhängige Informationen oder Informationen über Ereignisse. Zur Aktualisierung von dynamischen Informationen wird eine bestehende Netzverbindung benötigt. Durch Caching, Hoarding oder Replikation kann jedoch auch eine Offline-Nutzung zuvor heruntergeladener dynamischer Informationen ermöglicht werden. Es gibt zwei Subklassen dynamischer Informationen: (1) *Interne Informationen* sind Informationen, die zumeist vom Anbieter der App bereitgestellt werden. So werden in Zeitungs-Apps zum Beispiel aktuelle Nachrichten der zugehörigen Zeitung heruntergeladen und präsentiert. (2) *Externe Informationen* sind verknüpfte Informationen, die von Dritten bereitgestellt werden. Dies können zum Beispiel in Ereignisinformationen verknüpfte Videos bei Youtube sein, die nicht direkt vom Anbieter des Ereignisses zur Verfügung gestellt werden.

Bei der Konzeption einer App spielen auch Fragestellungen des Urheberrechts eine Rolle. In unserem Ansatz gehen wir davon aus, dass der Anbieter der App die nötigen Urheberrechte an den in einer App verwendeten statischen und internen Informationen besitzt. Ferner gehen wir davon aus, dass der Anbieter einer App bei der Verwendung von externen Informationen die Nutzungsbedingungen des jeweiligen Informationsanbieters berücksichtigt.

Generell ist davon auszugehen, dass Benutzer auf statische und dynamische Informationen nur lesenden Zugriff besitzen. Einige Apps unterstützen jedoch *interaktive* (dynamische) Informationen. So kann zum Beispiel eine App, die Informationen über Ereignisse bereitstellt, das Kommentieren von Ereignissen durch soziale Netzwerke wie Facebook, Twitter oder MySpace ermöglichen.

4. SYSTEMARCHITEKTUR UND PROOF-OF-CONCEPT

Aufgrund der besseren Verständlichkeit verzichten wir in diesem Papier auf die Trennung zwischen der Beschreibung unseres Ansatzes und der Implementierung des Proof-of-Concept-Systems namens MyBand-App. Dieses System ermöglicht es Musikern, ihre

eigenen Apps für Android- und iOS-basierte Smartphones zu erstellen. Uns ist bekannt, dass es mit FansMagnet[2] bereits einen Anbieter eines vergleichbaren Services für Musiker gibt. Jedoch werden dabei auch die Inhalte der Apps durch FansMagnet verwaltet. Aufgrund rechtlicher Bedenken, obliegt es in unserem Ansatz tatsächlich dem Informationsanbieter, dem Musiker oder dessen Beauftragten, sicherzustellen, dass alle Urheber- und Verwertungsrechtsfragen geklärt werden.

Der Proof-of-Concept stellt ein Anwendungsszenario von Bands bzw. Musikern dar, die Informationen an Ihre Fans über ein mobiles Informationssystem weitergeben möchten. Dieses Beispiel dient allerdings nur der Verdeutlichung unseres Ansatzes. Der hier beschriebene Ansatz lässt sich auf beliebige andere Informationsanbieter übertragen.

Um diese Übertragbarkeit zu gewährleisten, muss das System anpassbar und leicht erweiterbar sein. Aus diesem Grund ist unser System ist als Framework aufgebaut. Für die App-Erzeugung sind drei Komponenten verantwortlich: (1) ein Web-basiertes Konfigurationswerkzeug (MyBandServlet), (2) die Code-Skelette für die Apps und (3) die eigentliche App-Erzeugung. Der Arbeitsablauf beim Erstellen einer App ist wie folgt: Zuerst wählt der Nutzer die Zielplattform(en) und die Module, welche er/sie in die App integrieren will (vgl. Abbildung 1). Anschließend werden die App und die ausgewählten Module konfiguriert und der Erzeugungsprozess angestoßen. Nach dessen Beendigung (nach dem Kompilieren der App) kann der Nutzer seine App herunterladen.

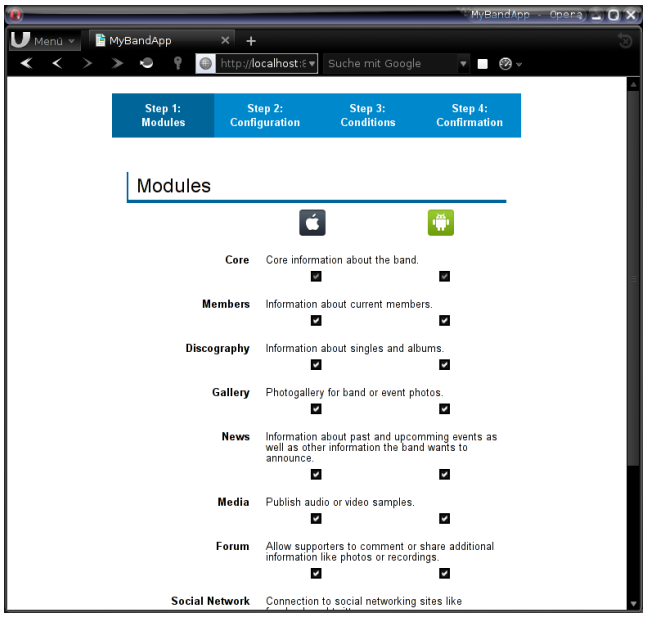

Abbildung 1: MyBand-App Web-basierte Konfiguration (Modulauswahl)

Aus einem technischeren Blickwinkel betrachtet, funktioniert unser Ansatz wie folgt: Wie Abbildung 1 verdeutlicht, sind Module die kleinsten Bausteine in unserem Erzeugungsprozess. Es gibt intern zwei Arten von Modulen: *BuilderModule* sind für die Erstellung des plattformabhängigen Binärcodes zuständig. *AppModule* kapseln die Funktionalität der App. Jedes AppModule umfasst das entsprechende Code-Skelett und eine XML-Datei, mit der die

[2]http://www.fansmagnet.com

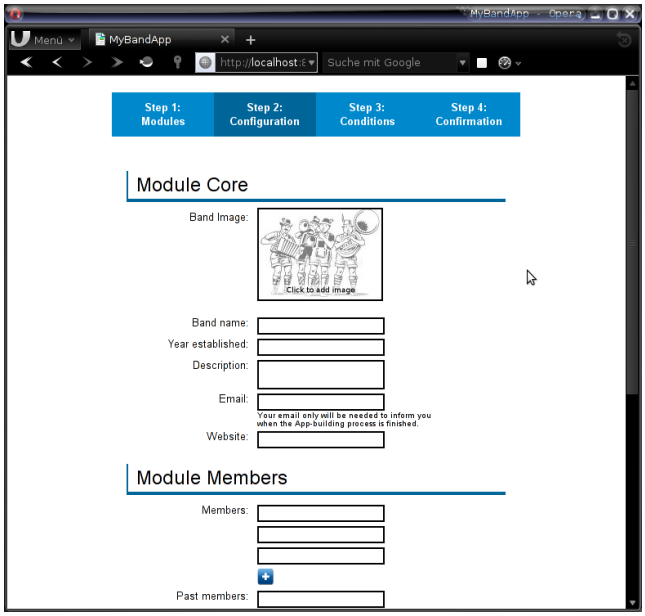

Abbildung 2: MyBand-App Web-basierte Konfiguration (Modulkonfiguration)

durch den Web-basierten Konfigurator festgelegte Modulkonfiguration (bestehend aus statischen und dynamischen Informationen) verarbeitet wird (vgl. Abbildung 2). Dies umfasst im verwendeten Anwendungsszenario unter anderem den Namen und das Bild der Band bzw. Musiker, sowie die Liste der Band-Mitglieder. Für die Diskographie werden z.B. die Datei `DiscographyModule.xml` und zwei AppModule `IPhoneDiscographyModule.jar` und `AndroidDiscographyModule.jar` genutzt. Bevor der eigentliche Erzeugungsprozess gestartet wird, analysiert das MyBand-Servlet die verfügbaren XML-Dateien, erzeugt aus den Meta-Informationen zu statischen Informationen die Formulare für den Konfigurator und generiert aus den Meta-Informationen zu dynamischen Informationen eine RSS-Vorlage[3], die zur späteren Bereitstellung der dynamischen Informationen dient. Nachdem das Konfigurationsformular durch den Nutzer abgeschickt wurde, wird pro App-Modul die `configure`-Methode, welche jedes AppModul implementiert, aufgerufen. Diese Methode modifiziert das Code-Skelett des entsprechenden AppModuls entweder direkt oder, wie in Abbildung 4 dargestellt, durch Ändern der XML-Datei basierend auf der durch den Nutzer angegebenen Konfiguration. Anschließend werden die konfigurierten AppModule an das BuilderModule übergeben, welches für das Kompilieren der App verantwortlich ist.

Die Abbildungen 5 und 6 zeigen eine iOS-App, welche beispielhaft mit dem beschriebenen MyBand-App-Ansatz erzeugt wurde. Sicherlich ist die dargestellte Oberfläche noch nicht „fancy“, die Abbildungen verdeutlichen aber, dass ein Java-basiertes Framework genutzt werden kann, um Programme in Objective-C zu erzeugen.

Abbildung 3 verdeutlicht, dass der vorgestellte Ansatz zum Ziel hat, so generisch wie möglich zu sein. Zur Unterstützungen von Apps abseits der Musiker-Domäne müssen lediglich andere AppModule und XML-Dateien bereitgestellt werden.

[3]Im Prototyp werden die RSS-Dateien händisch editiert. Ein Editor zum formularbasierten Eingeben der dynamischen Daten ist indes in Vorbereitung.

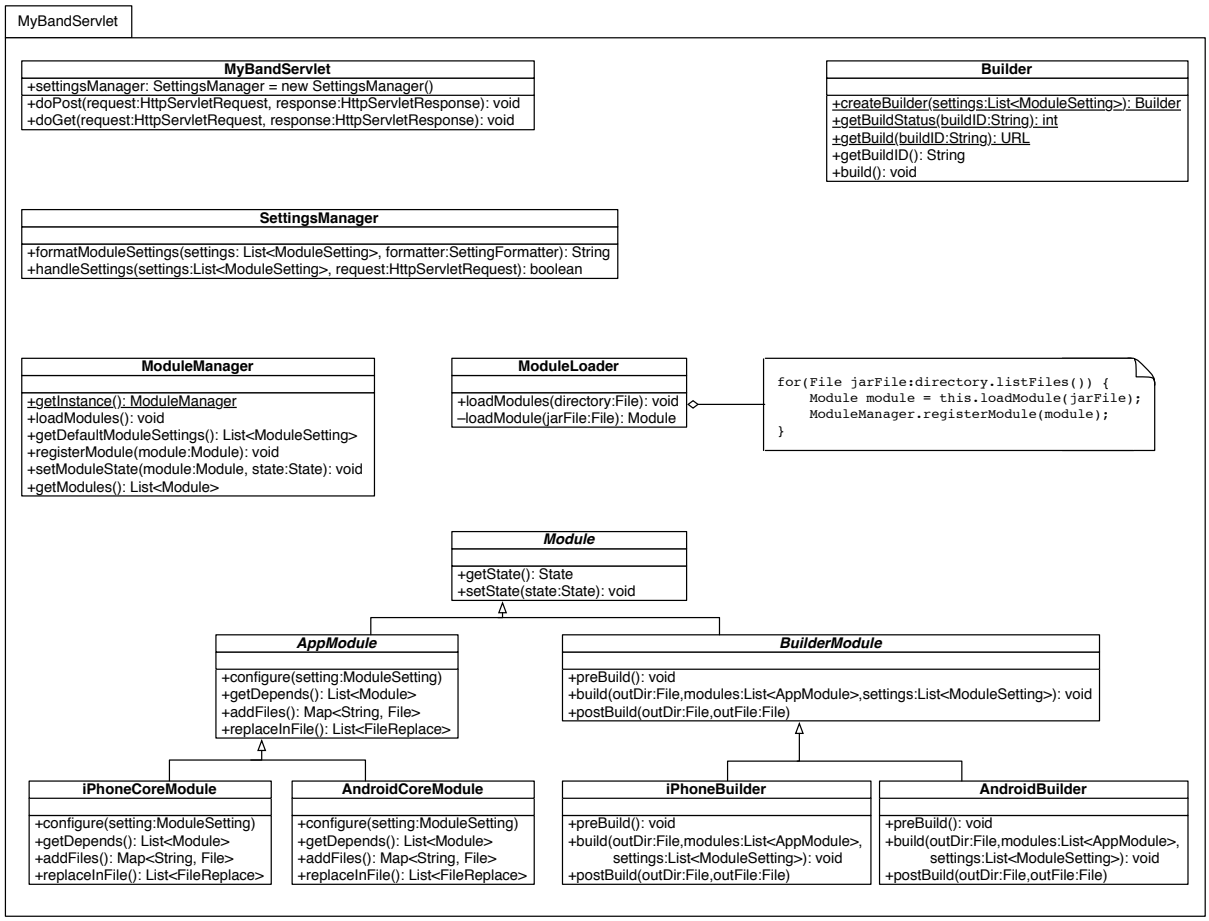

Abbildung 3: Klassendiagramm des Framework-Servlets für die MyBand-App-Konfiguration und -Erzeugung

Abbildung 4: Beispiel für die XML-basierte Transformation eines Android-Code-Skeletts

5. ZUSAMMENFASSUNG UND AUSBLICK

In diesem Beitrag haben wir erste Ideen präsentiert, die es Endbenutzern ohne technische Kenntnisse ermöglichen, Informationssystem-Apps für Smartphones verschiedener Betriebssysteme zu erstellen. Wir haben die generellen Eigenschaften von Informationen, die in solchen Apps angeboten werden, klassifiziert und diese Klassifikation für den Entwurf von Code-Skeletten verwendet. Mit Hilfe eines Web-Interfaces können Benutzer ihre App parametrisieren. Anschließend wird die App über die nativen Entwicklungswerkzeuge kompiliert und kann heruntergeladen werden. Weitere Vorzüge unseres Systems sind die native Benutzeroberfläche und die Berücksichtigung des Urheberrechts der angebotenen Informationen. Neben diesen allgemeinen Ansätzen haben wir unseren Prototypen „MyBand-App“ vorgestellt, den wir zudem einer breiten Öffentlichkeit auf der CeBIT-Messe im März 2011 in Hannover zeigen konnten.

Wir stehen erst am Anfang unserer Forschung und sehen nun verschiedene Forschungsrichtungen, die sich in Zukunft eröffnen. Zunächst werden wir interaktive Informationen berücksichtigen. Zudem werden wir die in unserem Evaluationssystem umgesetzten Ideen generalisieren, um die Erstellung von sehr unterschiedlichen mobilen Informationssystem-Apps zu unterstützen. Ferner werden wir somit die Formalisierung unseres Ansatzes beginnen, um die dem Ansatz bislang noch fehlende fundierte formale Basis zu erhalten.

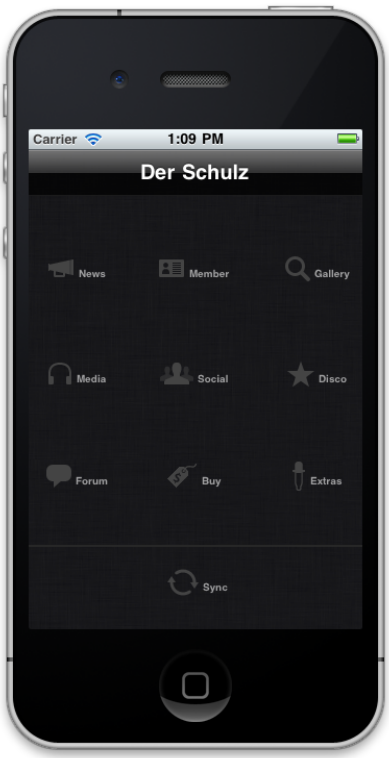

Abbildung 5: Beispiel: MyBand-App für iOS

Abbildung 6: Beispiel: MyBand-App für iOS (die Galerie)

6. LITERATURVERZEICHNIS

[1] N. Balasubramanian, A. Balasubramanian, and A. Venkataramani. Energy consumption in mobile phones: a measurement study and implications for network applications. In *Proceedings of the 9th ACM SIGCOMM conference on Internet measurement conference*, pages 280–293, New York, NY, USA, Nov. 2009. ACM.

[2] K. Czarnecki and U. Eisenecker. *Generative Programming: Methods, Tools, and Applications*. Addison-Wesley Professional, 2000.

[3] L. M. Feeney and M. Nilsson. Investigating the energy consumption of a wireless network interface in an ad hoc networking environment. In *Proceedings IEEE INFOCOM 2001, The Conference on Computer Communications, Twentieth Annual Joint Conference of the IEEE Computer and Communications Societies, Twenty years into the communications odyssey, 22-26 April 2001, Anchorage, Alaska, USA*, volume 3, pages 1548–1557, Los Alamitos, CA, USA, 2001. IEEE. available online: `http://www.sics.se/~lmfeeney/publications/Files/infocom01investigating.pdf`.

[4] H. Höpfner. Replication in Mobile Information Systems. In *Informatik bewegt*, volume P-19 of *Lecture Notes in Informatics (LNI)*, pages 590–593, Bonn, Germany, 2002. GI, Köllen Druck+Verlag GmbH.

[5] S. Kent. Model driven engineering. In M. Butler, L. Petre, and K. Sere, editors, *Integrated Formal Methods*, volume 2335 of *Lecture Notes in Computer Science*, pages 286–298. Springer Berlin / Heidelberg, 2002.

[6] J. Kjeldskov and C. Graham. A review of mobile hci research methods. In *Human-Computer Interaction with Mobile Devices and Services*, volume 2795 of *Lecture Notes in Computer Science*, pages 317–335. Springer Berlin / Heidelberg, 2003.

[7] G. H. Kuenning and G. J. Popek. Automated Hoarding for Mobile Computers. In W. M. Waite, editor, *Proceedings of the 16th ACM Symposium on Operating Systems Principles*, pages 264–275, New York, NY, USA, 1997. ACM Press.

[8] K. C. K. Lee, H. V. Leong, and A. Si. Semantic query caching in a mobile environment. *ACM SIGMOBILE Mobile Computing and Communications Review*, 3(2):28–36, 1999.

[9] P. Nicopolitidis, A. S. Pomportsis, G. I. Papadimitriou, and M. S. Obaidat. *Wireless Networks*. John Wiley & Sons, Inc., New York, NY, USA, 2003.

[10] J. Nielsen. ipad usability: First findings from user testing. Website, May 2010. `http://www.useit.com/alertbox/ipad.html`.

[11] C. Pettey. Gartner Says Worldwide Mobile Device Sales Grew 13.8 Percent in Second Quarter of 2010, But Competition Drove Prices Down. Website, Aug. 2010. `http://www.gartner.com/it/page.jsp?id=1421013`.

[12] Q. Ren and M. H. Dunham. Semantic Caching and Query Processing. *Transactions on Knowledge and Data Engineering*, 15(1):192–210, Jan. 2003.

[13] X. Wang. *Wired and Wireless Networks*. Vdm Verlag Dr. Müller, Saarbrücken, Germany, 2007.