Hochschule für Angewandte Wissenschaften Hamburg

Hamburg University of Applied Sciences

Ausarbeitung zur Veranstaltung „Projekt 2“ im Masterstudiengang Informatik WiSe 2010/11

Jan Schwarzer, Lorenz Barnkow

Entwicklung eines Multitouch-Informationssystems für öffentliche Wandbildschirme

Fakultät Technik und Informatik
Department Informatik

Faculty of Engineering and Computer Science
Department of Computer Science

Inhaltsverzeichnis

Abbildungsverzeichnis

Listings

1 Einführung

Diese Projektausarbeitung baut im Wesentlichen auf Vorarbeiten aus den vergangenen Semestern des Masterstudiums auf. Im Schwerpunkt fließen hier die Themenbereiche *Multitouch* (Barnkow (2010b,a)) und *Computer-Supported Cooperative Work* (Schwarzer (2010b,a)) inhaltlich mit ein. Hierbei wurden Kritikpunkte, Ideen und Anregungen aufgegriffen, um das im Folgenden beschriebene, praktische Beispiel in Form eines Feasiblity Prototypen[1] zu entwickeln. Fachlich lässt sich dieser Prototyp durch den Begriff der CommunityMirrors[2], so genannte Community-Unterstützungssysteme, beschreiben (Schwarzer (2010b)).

1.1 Motivation

Nach Koch und Toni (2004) gewinnen solche Community-Unterstützungssysteme bei der Entwicklung neuer Anwendungen von Informations- und Kommunikationstechnologien immer mehr an Bedeutung. Neben der Bereitstellung eines Mediums für die direkte Kommunikation und dem indirekten Austausch von Inhalten und Kommentaren, nennen Koch und Toni (2004) die Bereitstellung von Informationen über andere Mitglieder und die Unterstützung zum Finden potentieller Kommunikationspartner als wesentliche Grundaktivitäten in einer Community. Ziel der CommunityMirros ist es hierbei, ein Gewahrsein über einen bestimmten und gewünschten Sachverhalt, welcher wiederrum für den eigenen Kontext wichtig ist, zu schaffen (Dourish und Bellotti (1992)). Schlichter u. a. (1998) sehen das Vermitteln eines Gewahrseins als die größte Gemeinsamkeit aller Arten von Kollaborationsunterstützung.

Im 11. Stock des Gebäudes Informations- und Elektrotechnik[3] der Hochschule für Angewandte Wissenschaften Hamburg (HAW Hamburg)[4] besteht bereits ein Community-Unterstützungssystems, hier in Form eines Touch-fähigen Studierenden-Informationssystems[5], welches u. a. Funktionen wie Raumbelegungen, Termine oder Veranstaltungspläne zur Verfügung stellt.

Durch die im Folgenden vorgestellen Arbeiten soll dieses Informationssystem in verschiedener Hinsicht erweitert und optimiert werden. So soll beispielsweise sowohl eine benutzerfreundliche Oberfläche geschaffen, als auch zusätzliche Funktionen implementiert werden.

1.2 Aufbau

Neben dieser Einleitung ist diese Projektausarbeitung in zwei weitere inhaltliche Kapitel gegliedert. Kapitel 2 geht auf die konkreten Design- und Realisierungsschritte ein, welche im

[1] zu Deutsch: „Machbarkeits-Prototyp"
[2] http://www.communitymirrors.net/
[3] http://newsletter.haw-hamburg.de/12078.html?&L=2%22
[4] http://www.haw-hamburg.de/
[5] http://www.informatik.haw-hamburg.de/index.php?id=2473&no_cache=1

Zuge der Projektarbeit umgesetzt wurden. Kapitel 3 fasst alle wesentlichen Punkte noch einmal zusammen, verdeutlicht den aktuellen Stand der Entwicklung des Prototypen und weist ausblickend auf fortführende Arbeiten hin.

2 Design und Realisierung

2.1 Systemkomponenten

Im Folgenden sollen alle wesentlichen und umgesetzten Komponenten des Informationssystems vorgestellt werden. Dabei gilt es zu beachten, dass nicht alle Bestandteile des Informationssystems, im Rahmen der Veranstaltung *Projekt 2*, umgesetzt werden konnten. So fehlen bspw. die Rubriken *Raumbelgung* oder *Stockwerkansicht*.

Die umgesetzten Komponenten sind das *Basisframework*, die Rubrik *Mitarbeiter und Professoren* und eine *Ideenplattform*.

2.1.1 Basisframework

Zu Beginn der Entwicklung mussten erst Erfahrungen mit der Anwendungsentwicklung in WPF 4[6] gesammelt werden. Dies betraf sowohl die Gestaltung eines Systems aus mehreren Teilanwendungen sowie die Verarbeitung der Toucheingaben.

Im Gegensatz zur verbreiteten Desktop-Metapher[7], bei der der Bildschirm als Arbeitsfläche für eine Vielzahl von Dokumenten und Ordnern dient, fiel die Entscheidung auf eine einfachere Struktur. Angelehnt an die mobilen Systeme Android[8] oder iOS[9] soll im entwickelten Informationssystem immer nur eine Teilanwendung (View) gleichzeitig aktiv sein, die dann auch den gesamten Bildschirm belegen kann. Diese Entscheidung wurde getroffen, weil das System als Einzelbenutzeranwendung geplant war und um die Verwaltung der Views zu vereinfachen.

In der aktuellen Umsetzung werden die aktiven Views auf einem Stack verwaltet, wobei das oberste Stackelement die zurzeit aktive und sichtbare View darstellt. Ähnlich wie bei Android-Anwendungen, kann eine View eine weitere View anzeigen lassen, indem sie der Viewverwaltung dies mitteilt. Das Ein- und Ausblenden der vorherigen und der neuen View übernimmt dann die Viewverwaltung. Beim Austausch der einzelnen Views konnten die umfangreichen Animationsmöglichkeiten von WPF genutzt werden, um vielfältige Effekte zu erzielen. Die Gestaltung der Animationen konnte außerdem mit Werkzeugen durchgeführt werden, so dass fast kein Code manuell geschrieben werden musste. Eine exemplarische Animation ist in Abbildung 1 zu sehen.

[6] Windows Presentation Foundation – http://msdn.microsoft.com/en-us/library/ms754130.aspx
[7] http://en.wikipedia.org/wiki/Desktop_metaphor
[8] http://www.android.com/
[9] http://www.apple.com/ios/

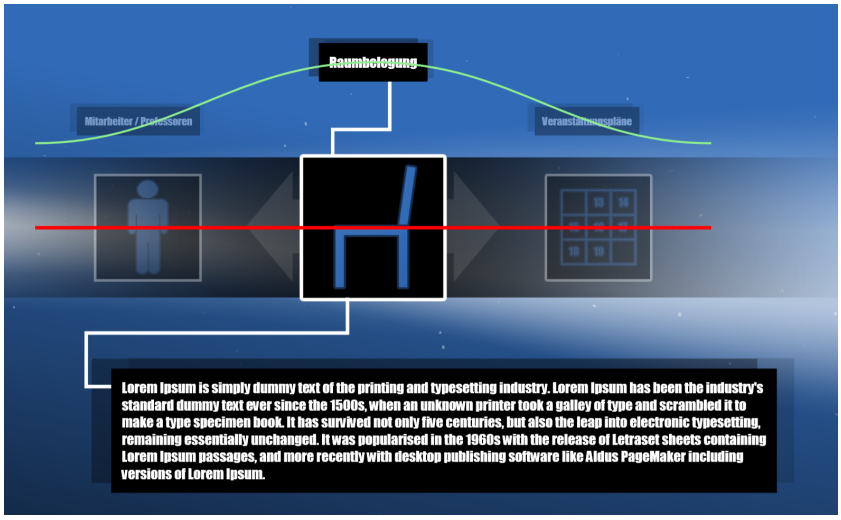

Abbildung 2: Ausschnitt des Hauptmenüs mit Animationspfaden

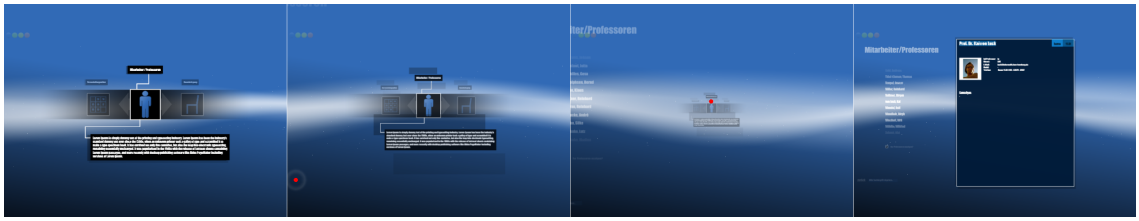

Abbildung 1: Animierter View-Wechsel vom Hauptmenü in die Mitarbeiter und Professoren-Suche

Das Start- bzw. Hauptmenü des Informationssystem orientiert sich grob an der sog. Surface Shell des Microsoft Surface (Version 1.0)[10]. Die einzelnen Teilanwendungen werden als Optionen auf einem unendlichen Band präsentiert und können mittels Touchgesten verschoben und aktiviert werden. Durch die Entwicklung eines solchen Widgets konnten die Kenntnisse von WPF 4 weiter vertieft werden.

Das entwickelte Menü ist in Abbildung 2 zu sehen und unterscheidet sich von der Surface Shell dahingehend, dass die Menüelemente nicht linear aufgereit sein müssen. Zur Positionierung der Elemente können Pfade vorgegeben werden, die im einfachsten Fall Geraden sein können. Es ist aber auch möglich, beliebig viele Bézierkurven zu einem Pfad zusammen zu setzen. In der Abbildung 2 ist zu sehen, dass die Grafiken entlang der roten Gerade dargestellt werden, während die Beschriftungen der grünen Kurve oben folgen.

Als letzte vorbereitende Arbeit musste die Datenhaltung für das Informationssystem behandelt werden. Aufgrund der Vorkenntnisse aus dem Studium an der HAW Hamburg, sollten die meis-

[10] http://www.microsoft.com/surface/

ten Daten in einer relationalen Datenbank gehalten werden. Um eine Flexibilität hinsichtlich der Wahl des Datenbanksystems zu gewährleisten, sollte die Anbindung der Datenbank in der Anwendung über die Bibliothek NHibernate[11] erfolgen. NHibernate ist ein ORM[12]-Framework mit der Aufgabe die Domänenobjekte der Anwendung auf relationale Tabellen abzubilden und umgekehrt. Der Kern der Anwendung wird durch Konfigurationsdateien, die dieses Mapping beschreiben, von den speziellen Anforderungen der verschiedenen Datenbankanbieter befreit. So können für die einzelnen Teilanwendungen des Informationssystems jeweils Domänenmodelle unabhängig von der darunterliegenden Datenbank erstellt werden.

2.1.2 Mitarbeiter und Professoren

Eine nach Koch und Toni (2004) grundlegende und wichtige Eigenschaft, welche ein CommunityMirror erfüllen sollte, ist das so genannte *Matchmaking*. Das Matchmaking beschreibt im Kern das zielgerichtete Finden von Kommunikationspartnern. Im Kontext dieser Projektausarbeitung bezieht sich das Matchmaking auf das Finden von Informationen zu Mitarbeitern und Professoren. Eine solche Information kann u.a. eine E-Mail-Adresse, eine Telefonnummer oder auch eine Raumnummer sein.

Im ersten Schritt wurde das Gebäude Informations- und Elektrotechnik der HAW Hamburg kartographiert. Dazu wurden alle Stockwerke, Räume, Personen in diesen Räumen sowie angegebene zusätzliche Informationen erfasst. Die Informationen wurde XML[13]-basiert gespeichert.

Im Kern gliedert sich die XML-Datei in drei wesentliche Komponenten:

1. **Stockwerke:** Ein *Stockwerk* besitzt die Attribute *Titel*, *Titelzusatz*, *Nummer* und eine Liste *Raeume*. In Raeume wird deutlich welcher *Raum* über eine *Id* einem *Stockwerk* zugewiesen wurde.
2. **Räume:** Ein *Raum* bestitzt die Attribute *Id*, *Titel*, *Beschreibung*, eine Liste *Personen* und *Sonstiges*. In *Personen* wird deutlich, welche *Person* über eine *Id* einem *Raum* zugewiesen wurde. Sonstiges lässt Platz für weitere Angaben, wie bspw. Öffnungszeiten, Kontaktinformationen, etc.
3. **Personen:** Eine *Person* besitzt die Attribute *Id*, *Titel*, *Vorname*, *Nachname*, *IstProfessor*, *Kuerzel*, *Foto*, eine Liste *Telefonnummern*, *Mobil*, *E-Mail* und *Sonstiges*. Dabei identifiziert *IstProfessor*, ob eine *Person* ein Professor oder ein wissenschaftliche Mitarbeiter bzw. sonstiger Angestellter ist. *Telefonnummern* enthält alle *Telefonnummern*, die eine *Person* besitzt und gibt mithilfe einer *Id* eines *Raums* darüber Aufschluss, wo diejenige *Person* unter der angegebenen *Telefonnummer* zu erreichen ist.

[11] http://nhforge.org/Default.aspx
[12] Object-Relational Mapping
[13] Extensible Markup Language – http://www.w3.org/TR/2008/REC-xml-20081126/

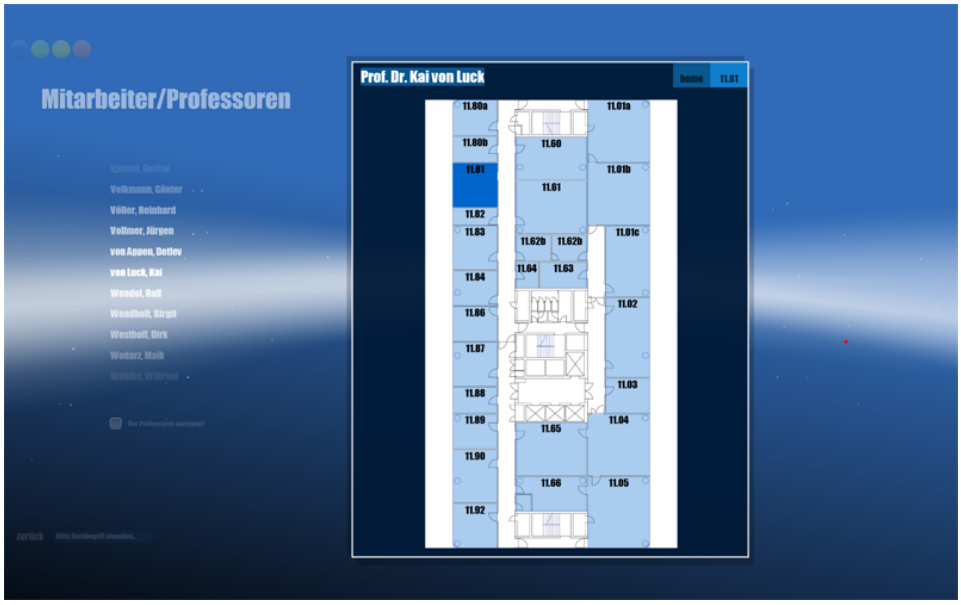

Abbildung 3: Oberfläche für das Matchmaking

Im zweiten Schritt wurde die Komponente für die Darstellung im Prototypen entwickelt (siehe Abbildung 3). Die Oberfläche gliedert sich in drei Abschnitte:

1. **Navigation:** Die Navigation enthält sämtliche Vor- und Nachnamen aller Personen aus der zuvor genannten XML-Datei. Sie kann nach dem Personen-Attribut *IstProfessor* gefiltert werden. D.h. der Anwender kann entscheiden, ob er nur die Vor- und Nachnamen von Professoren angezeigt bekommt oder eben die Vor- und Nachnamen aller Personen. Die Bedienung erfolgt über vertikale Touch-Bewegungen, wodurch die Navigation zum Durchscrollen der Liste der angezeigten Personen aufgefordert wird.
2. **Suche:** Zusätzlich hat der Anwender die Möglichkeit, per Software-Tastatur konkret nach Personen zu suchen. Die Navigation aktualisiert sich parallel zu den Eingaben in dem Suchfeld und zeigt so immer die zur Zeit zutreffenden Suchtreffer an. Solange das Software-Keyboard aktiv ist, bleibt die Darstellungsmaske ausgeblendet.
3. **Darstellung:** In Abbildung 3 ist die Ansicht eines konkreten Profils zu sehen. Es existieren konkret zwei Ansichts-Masken für Personenprofile. Zum einen die Profilansicht selbst, in welche alle zuvor beschriebenen Personen-Attribute visualisiert werden und zum anderen eine Stockwerkansicht, in welcher zu sehen ist, in welchem Stockwerk sich eine Person i .d. R. aufhällt aber auch, in welchem Raum diese sich befindet.

Ziel bei der Entwicklung dieser Komponente war es, den Studierenden eine einfache und schnelle Möglichkeit zu bieten, Informationen von gewünschten Ansprechpartnern zu finden, ohne dabei in der Funktion zu komplex zu werden.

2.1.3 Ideenplattform

Eine sehr umfangreiche Komponente des Informationssystems ist die Ideenplattform, welche im Folgenden im Detail beschrieben werden soll.

2.1.3.1 Konzept In Schwarzer (2010b) wurde das Konzept des so genannten *IdeaMirrors*, eine spezielle Form des CommunityMirrors, und dessen konkreter Einsatz im Rahmen des Forschungsprojektes Gemeinschaftsgestützte Innovationsentwicklung für Softwareunternehmen (GENIE)[14] vorgestellt. Ein IdeaMirror verfolgt primär das Ziel, das kreative Ideenpotential eines Unternehmens sichtbar zu machen, um so die Ideengenerierung und kooperative Ideenvernetzungen zu fördern (Koch und Möslein (2007)). In diesem Kontext wird in dem Forschungsprojekt GENIE versucht, die Innovationsfähigkeit von Unternehmen durch so genannte *Innovationscommunities*, konkret Communities, in welchen Mitarbeiter, Kunden oder Projektpartner beteiligt sind, günstig zu beeinflussen (Schwarzer (2010b)). Die Ideen wurden dabei durch die Teilnehmer dieser Innovationscommunities generiert und durch einen entsprechenden Ideenevolutions- und Ideenevaluationsprozess erstellt und bewertet. Der praktische Kontext war dabei eine Studie, die bei dem Softwareunternehmen SAP[15] unter dem Name *SAPiens* durchgeführt wurde (Blohm u. a. (2010)). Neben einer Web-Oberfläche, für teilweise administrative Arbeiten, wurden mithilfe des Ideamirrors die verschiedenen Ideen visualisiert und auf einem interaktiven Wandbildschirm präsentiert (Blohm u. a. (2010), Ott (2010)).

Einen einfacheren aber ähnlichen Ansatz verfolgt Youtube[16] mit seinem neuen Kommentar-Bewertungssystem. Im Jahre 2010 vereinfachte Youtube dieses Bewertungssystem, indem die Anwender nur noch einen Beitrag mit einem *gefällt mir* oder *gefällt mir nicht* bewerten können. Auch können einzelne Kommentare durch entsprechende Bewertungen auf- bzw. abgewertet werden. Bspw. hat dieses zur Folge, dass oft gut bewertete Kommentare direkt unterhalb des Videos zu sehen sind. Es stehen also immer die Kommentare ganz oben, die die meisten positiven Kommentar-Bewertungen erhalten haben. Diese Funktion nennt sich in Youtube *Highlightsanzeige* (InternetWorld (2010)).

Ein ähnliches Konzept wurde in dem hier beschriebenen Prototypen implementiert. Neben dem einleitend beschriebenen Gewahrsein, spielt nach Koch und Möslein (2007) die Wertschätzung auf Ideenplattformen eine weitere und wichtige Rolle. D. h. beigetragende Ideen eines Community-Migliedes sind klar und deutlich auch als eben diese erkennbar. Ziel ist es hierbei, die Motivation für das Mitteilen von eigenen Ideen zu steigern (Koch und Möslein (2007)). Der Kontext, in dem hier vorgestellten Informationssystem, ist hingegen offen. Ideen sind nicht auf Themengebiete eingeschränkt (vgl. Blohm u. a. (2010)). Bspw. können sich die Ideen und

[14] http://projekt-genie.de/
[15] http://www.sap.com/
[16] http://www.youtube.com/

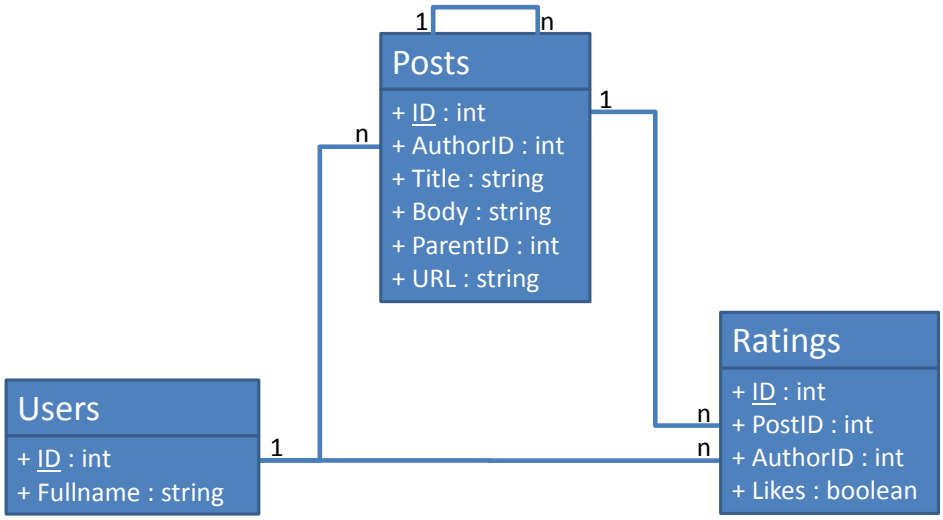

Abbildung 4: Klassendiagramm der Ideenplattform

Kommentare mit Themen wie das Mensaessen, Studenplänen, Öffungszeiten, etc. beschäftigen. Im Kern zählt, dass eine Idee veröffentlicht und anschließend bewertet werden kann.

Eine weiterer wesentlicher Bestandteil der hier vorgestellten Ideenplattform ist das Einbinden von bereits bestehenden so genannten Social Network Services (SNS)[17]. Dazu zählen u. a. die hier im Folgenden beschriebenen und integrierten Komponenten von Facebook[18] und Twitter[19] aber auch zusätzliche Anbindungen an andere SNS. Für den Kontext dieses Prototypen wurde der Integrationsaufwand auf die beiden zuerst genannten SNS beschränkt. Ausschlaggebend war hierbei vor allem die sehr hohe Verbreitung von Facebook. Laut der Webseite Alexa (2011)[20] ist Facebook die weltweit am zweithäufigsten aufgerufene Webseite überhaupt.

Durch die Integration dieser SNS, soll es dem Anwender ermöglicht werden, sich in seiner gewohnten digitalen Umgebung aufzuhalten, dabei aber gleichzeitig ihm auch die Möglichkeit zu geben, Ideen oder Kommentare für die Ideenplattform zur Verfügung zu stellen.

2.1.3.2 Fachliches Modell Die drei wesentlichen Bestandteile des fachlichen Modells der Ideenplattform – die schriftlichen Beiträge (Klasse *Posts*), die Benutzer (Klasse *Users*) und die Bewertungen (Klasse *Ratings*) – sind in Abbildung 4 abgebildet.

Authentifizierte Benutzer haben eine eindeutige Identifikationsnummer (Feld *ID*) und einen Namen (Feld *Fullname*), unter dem sie beliebig viele Beiträge veröffentlichen können.

[17] http://en.wikipedia.org/wiki/Social_network_service
[18] http://en.wikipedia.org/wiki/Social_network_service
[19] http://twitter.com/
[20] http://www.alexa.com/

Die Attribute der Beiträge setzen sich aus einer eindeutigen Identifikationsnummer (Feld *ID*), dem zugeordneten Benutzer (Feld *AuthorID*), der Überschrift (Feld *Title*) und dem Beitragstext (Feld *Body*) sowie einem Hierarchieverweis (Feld *ParentID*) und einer URL zusammen. Sowohl neue Ideen, als auch die Antworten bzw. Kommentare anderer Benutzer darauf, werden im fachlichen Modell als Beiträge modelliert. Bei einer Idee ist das Feld *ParentID* nicht gesetzt, während Antworten bzw. Kommentare jeweils auf die *ID* des zugeordneten Beitrags verweisen. Eine Begrenzung der Hierarchietiefe ist hierbei nicht vorgesehen. Das Feld *URL* dient zum Aufruf des Beitrags aus dem Internet, wird zurzeit jedoch nicht verwendet, da eine entsprechende Infrastruktur im Rahmen der Veranstaltung *Projekt 2* nicht realisiert wurde.

Die Möglichkeit der Bewertung der Beiträge durch die Benutzer ist in der Klasse *Ratings* umgesetzt. Durch die Felder *PostID* und *AuthorID* kann jeder Benutzer jeden Beitrag positiv oder negativ bewerten (Feld *Likes*).

Dieses Modell wurde im Rahmen des Projekts mit Microsoft Access 2007[21] in eine relationale Datenbank überführt. Durch die Verwendung des ORM-Frameworks NHibernate (siehe Abschnitt 2.1.1) wäre ein Austausch gegen ein leistungsfähigeres RDBMS[22] zu einem späteren Zeitpunkt problemlos möglich. Die so genannten Mappings befinden sich in Anhang A.2.

2.1.3.3 Integration externer Plattformen Wie im einleitenden Konzept zur Ideenplattform beschrieben (Abschnitt 2.1.3.1), war die Einbindung externer SNS von Beginn an geplant. Hierbei werden die Beiträge in beide Richtungen synchronisiert, so dass neue Ideen, die am Wandbildschirm erfasst wurden, automatisch über die entsprechenden SNS veröffentlicht werden und Beiträge aus diesen SNS auch wieder in die Datenbank der Ideenplattform einfließen (siehe Abbildung 5). Als Kommunikations-Schnittstelle dient dabei das in Abschnitt 2.1.3.2 beschriebene fachliche Modell in der Ideenplattform.

Zur Realisierung wurde die Datenbank um weitere Tabellen erweitert, um die externen Daten zwischenzuspeichern und eine Zuordnung zu den Daten des fachlichen Modells herzustellen. Die entsprechenden Mappings für NHibernate sind in Anhang A.3 und A.4 abgedruckt. Die Synchronisation wird zurzeit von einem einzelnen Thread durchgeführt, der periodisch die Synchronisationsfunktionen der Twitter- und Facebook-Anbindung aufruft.

Die Einzelheiten zu den verwendeten Schnittstellen können den beiden folgenden Abschnitten entnommen werden.

2.1.3.4 Twitter-Anbindung Eine, der beiden hier implementierten SNS-Komponenten, ist Twitter. Twitter ist nach Wikipedia (2011b) eine Anwendung für das so genannte *Mikroblogging*.

[21] http://office.microsoft.com/de-at/access/
[22] Relationales Datenbankmanagementsystem.

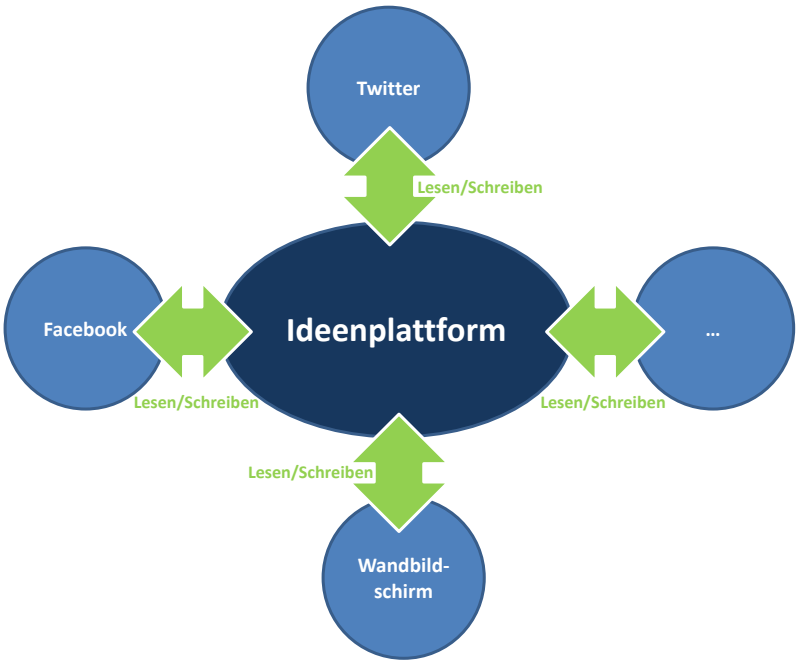

Abbildung 5: Zusammenspiel der Komponenten der Ideenplattform

Mit Mikroblogging wird eine Form des Bloggens beschrieben, welche nur eine sehr begrenzte Größe für die zu verfassenden Kurznachrichten zulässt (Wikipedia (2010b)). Im Falle von Twitter ist eine Kurznachricht auf 140 Zeichen beschränkt (Twitter.com (2011)).

Das zugrunde liegende Kommunikationsmodell von Twitter sieht folgende Aspekte vor (Twitter.com (2011)):

- Jeder Benutzer besitzt ein eigenes Profil, in welchem dieser Kurznachrichten auf seiner eigenen so genannten *Timeline* verfassen kann.
- Andere Benutzer haben die Möglichkeit, als so genannte *Follower*, dem Profil eines konkreten Benutzers zu folgen. D. h. sie erhalten laufend die aktuellen Nachrichten aus dem Profil des Benutzers.
- Außerdem besitzt jeder Benutzer die Möglichkeit selbst anderen Benutzern zu folgen. Hier spricht man vom so genannten *Following*.

Zwei weitere wichtige Kommunikationskomponenten sind die so genannten *Retweets* und *@Antworten* (Twitter.com (2011)). Ersteres beschreibt im Kern das Weiterleiten einer für einen Benutzer spannenden Kurznachricht an alle seiner *Follower*. @Antworten sind öffentliche Antworten, haben das Format *@Twitter-Benutzername* und unterscheiden sich von normalen Antworten vor allem dadurch, dass ein Benutzer nicht zwingend den betroffenen Benutzer als *Follower* bekannt sein muss. Eine weitere Restriktion für @Antworten ist es, dass diese im-

mer am Anfang einer Nachricht zu stehen haben, andernfalls wird die @Antwort nur als eine Erwähnung interpretiert.

Auf Basis dieses Konzepts wurde eine Twitter-Komponente mithilfe der Twitter-API[23] für das Informationssystem entwickelt. Zu Testzwecken wurden mehrere Twitter-Benutzerkonten angelegt. Der zentrale Benutzeraccount *hawmirror* diente dabei als Basis für das Veröffentlichen von Nachrichten auf Twitter, parallel aber auch als Schnittstelle für die Ideenplattform. Dabei spielt die folgende Idee eine wesentliche Rolle: Ein Twitter-Benutzer kann seine Ideen auf Twitter für die Ideenplattform veröffentlichen, indem entweder der Benutzer *hawmirror* diesen Twitter-Benutzer als *Following* kennt oder der Twitter-Benutzer eine @*hawmirror* Nachricht verwendet.

Es gibt allerdings eine Besonderheit für @Antworten: Bekommt der Benutzer *hawmirror* über eine @Antwort eine Idee oder ein Kommentar mitgeteilt wird dieser nicht auf der Timeline des Benuters *hawmirrors* angezeigt. Dort sieht Twitter eine eigene Timeline, welche sich @*Erwähnungen* nennt, vor. Da der Benutzer der diese Idee oder den Kommentar verfasst hat, durchaus dem Benutzer *hawmirror* ungekannt ist, muss diese @Erwähnung vorerst über ein *retweet* an alle Follower des Benutzers *hawmirror* versendet werden, wodurch die @Erwähung gleichzeitig auf der Timeline des Benutzers *hawmirror* erscheint. Dieser Sonderfall ist jedoch im Kontext des Informationssystems nicht relevant, da hier davon ausgegangen wird, dass Benutzer sich explizit gegenüber dem Informationssystem mit ihrem Twitter-Account erkenntlich machen.

Unter Berücksichtigung dieser Aspekte ist es möglich sowohl Ideen und Kommentare aus Twitter heraus in die Ideenplattform und dadurch auch in Facebook zu übertragen aber auch aus der Ideenplattform oder Facebook Ideen und Kommentare in Twitter zu veröffentlichen.

2.1.3.5 Facebook-Anbindung Mit Facebook kann über eine Vielzahl von Protokollen, APIs[24] und Bibliotheken interagiert werden. Die wichtigsten Schnittstellen sind die so genannte Graph API[25] und die Facebook Query Language (FQL)[26]. Beide Schnittstellen können verwendet werden, um Anfragen per HTTP[27] mit entsprechenden URLs und Parametern zu stellen. Die Ergebnisse werden dann als JSON[28]- oder XML-Daten zurückgeliefert.

Mit Hilfe der Graph API können die einzelnen Objekte innerhalb von Facebook aufgesucht und anhand ihrer Beziehungen zu benachbarten Objekten navigiert werden. Beispielsweise kann, ausgehend von einem Benutzer, zu Freunden navigiert werden und von dort aus lassen sich alle Beiträge der Freunde auffinden. Problematisch ist an dieser Stelle, dass keine gezielten

[23] http://apiwiki.twitter.com/w/page/22554648/FrontPage
[24] Application Programming Interface
[25] http://developers.facebook.com/docs/reference/api
[26] http://developers.facebook.com/docs/reference/fql
[27] Hypertext Transfer Protocol – http://tools.ietf.org/html/rfc2616
[28] JavaScript Object Notation – http://tools.ietf.org/html/rfc4627

Abfragen über Zeiträume und keine Sortierung durchgeführt werden kann. Nur Beiträge abzufragen, die z. B. durch Kommentare anderer ergänzt wurden, ist nicht möglich. Es müssten alle Beiträge und ihre Kommentare einzeln abgefragt werden, um Aktualisierungen zu entdecken.

Beim FQL-Ansatz werden die Facebook-Daten logisch als (relationale) Tabellen betrachtet und können über eine SQL[29]-ähnliche Sprache abgefragt werden. Diese ist jedoch in vielerlei Hinsicht eingeschränkt, so dass u. a. das kartesische Produkt oder der Verbund mehrerer Tabellen (JOIN) nicht erlaubt ist. In einigen Fällen kann dies jedoch über Unterabfragen umgangen werden. Die Abfrage, um alle Beiträge der Pinnwand eines Benutzers zu erhalten, die sich nach einem bestimmten Zeitpunkt verändert haben und aufsteigend nach Erstellungsdatum sortiert sind, ist in Listing 1 zu sehen.

Listing 1: Exemplarische FQL-Abfrage

```
1 SELECT post_id, actor_id, created_time,
2        updated_time, message
3 FROM stream
4 WHERE source_id = <<userid>>
5 AND updated_time > <<timestamp>>
6 ORDER BY created_time ASC
```

2.2 Authentifizierung der Benutzer

Für die Authentifizierung der Benutzer am Informationssystem wurden mehrere Möglichkeiten in Betracht gezogen. Die einfachste Lösung stellt hierbei die Authentifizierung über Eingabe eines Benutzernamens und Passworts dar. Diese Eingabe könnte über eine Bildschirm- oder Software-Tastatur erfolgen. Die Untersuchungen von Sears (1991) zeigen jedoch eine Halbierung der durchschnittlichen Tippgeschwindigkeit auf. Deshalb wurden alternative Mechanismen gesucht und teilweise experimentell erprobt, um den Vorgang der Authentifizierung für den Benutzer so schnell und reibungslos wie möglich zu gestalten.

2.2.1 RFID

Eine der hier beispielhaft angewandten Authentifizierungs-Verfahren, ist die Authentifizierung über Radio-Frequency Identification (RFID)[30]. RFID ist nach Plötz (2008) ein Oberbegriff für Identifikationssysteme über Funkkanäle und baut auf zwei wesentliche Komponenten auf. Zum

[29] Structured Query Language – http://www.iso.org/iso/catalogue_detail.htm?csnumber=45498

[30] http://de.wikipedia.org/wiki/RFID

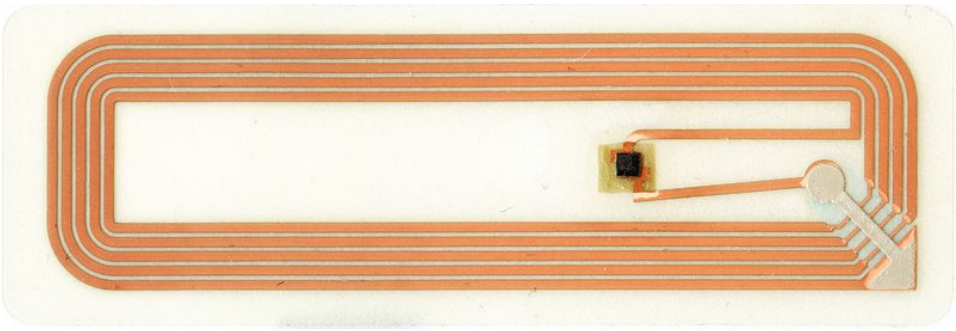

Abbildung 6: Interner Aufbau eines RFID-Transponders aus Wikipedia (2011a)

einen den so genannten *RFID-Transponder* und zum anderen die *RFID-Station* (Wikipedia (2011a)).

Im Kontext dieser Projektausarbeitung wird sich auf passive Systeme beschränkt, also Systeme, bei denen der RFID-Transponder seine Energie von der RFID-Station bezieht (Plötz (2008)). RFID-Transponder gibt es in unterschiedlichsten Formen, wie bspw. im Checkkarten-Format, als Schlüsselanhänger, etc. Hauptaufgabe des RFID-Transponders ist es, personenbezogenen Daten an die RFID-Station bei Bedarf zu übermitteln. Die RFID-Station dient dem eigentlichen Auslesen der Informationen aus dem permanenten Speicher des RFID-Transponder. Die RFID-Station baut dazu ein elektromagnetisches Feld auf, wodurch die im RFID-Transponder enthaltende Spule aufgeladen wird und den Chip mit der nötigen Spannung versorgt (siehe Abbildung 6). Der RFID-Transponder wird dadurch veranlasst, seine Seriennummer und weitere angeforderte Informationen der RFID-Station zuzusenden.

Alle Studierenden der HAW Hamburg besitzen eine RFID-Karte auf Basis der so genannten Mifare-Architektur (Wikipedia (2011b)). Mifare ist ein Produkt des Unternehmens NXP Semiconductors[31]. Konkreter wird bei den Studierenden-Ausweisen ein Mifare DESfire Chip verwendet, der im Gegensatz zu den Standardvarianten der Mifare-Architektur (vgl. Wikipedia (2011b)), eine Verschlüsselung mit sich bringt (Philips (2004)). Die Abbildung 7 zeigt die Komponenten der Mifare-Architektur noch einmal im Überblick.

Der Mifare DESfire Chip arbeitet auf Basis des ISO/IEC Standards 14443 (Wikipedia (2011a)). Dieser Standard definiert u. a. die physikalischen Eigenschaften der Endgeräte, das Protokoll, mit welchem Daten zwischen RFID-Transponder und RFID-Station ausgetauscht werden oder auch die Übertragungsfrequenz. Im Falle des Studierenden-Ausweises wird auf eine kurzwellen Frequenz von 13,56MHz gesetzt (Wikipedia (2011a)), wodurch die maximale Reichweite des Mifare DESfire Chips auf 100mm begrenzt ist (Philips (2004)).

Ein weiterer wichtiger Aspekt des Mifare DESfire Chips ist seine sieben Byte lange eindeutige Identifikationsnummer, anhand derer jede Mifare-Karte eindeutig identifiziert werden kann

[31] http://www.nxp.com/

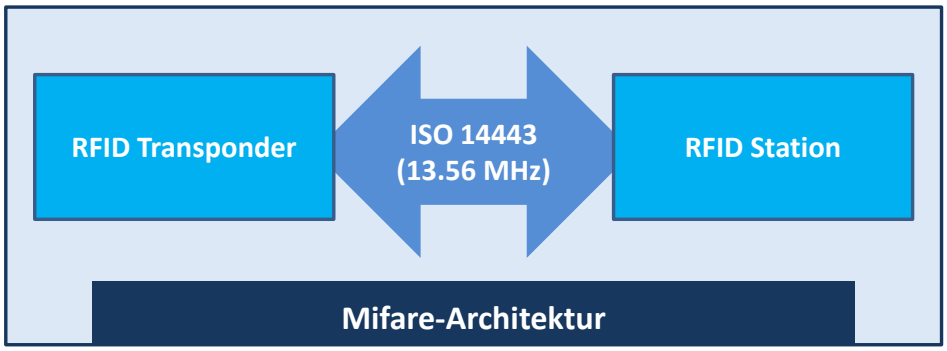

Abbildung 7: Komponenten der Mifare-Architektur

(Philips (2004)). Diese Identifikatinsnummer wird beim Auslesen des Mifare DESfire Chips periodisch der RFID-Station mitgeteilt.

Ausgelesen wird der Mifare DESfire Chip mithilfe eines RFID-Readers. Konkret wurde dazu das Gerät OMNIKEY 5513 Reader Board Compact[32] für ca. 36,00 EUR eingesetzt. Dieser Reader wurde über seine seriellen Anschlüsse mit dem Computer verbunden (TAGnology (2008)). Sofern ein Studierenden-Ausweis an den Reader gehalten wurde, überträgt dieser die sieben Byte lange Seriennummer an das Informationssystem, wodurch Rückschlüsse auf den Besitzer des Studierenden-Ausweises gezogen werden können.

Der Vorteil dieses Authentifizierungsverfahrens ist vor allem die Schnelligkeit aber auch die kontaktlose Authentifizierung am System selbst. Daher wurde eine Authentifizierung mit Hilfe von RFID an dem hier beschriebenen Informationssystem umgesetzt.

2.2.2 Gesichtserkennung

In der Einführung in die biometrische Erkennung von Personen, bewerten Jain u. a. (2004) die vorgestellten Verfahren nach verschiedenen Kriterien. Die bildbasierte Gesichtserkennung zeichnet sich dadurch aus, dass sie grundsätzlich auf alle Menschen anwendbar und leicht zu erfassen ist sowie über eine hohe Benutzerakzeptanz verfügt. Gegen diese Technik sprechen jedoch teils hohe Ähnlichkeiten zweier Gesichter, eingeschränkte Invarianz der Gesichter über die Zeit, hohe Rechenaufwände und die starke Anfälligkeit für Betrugsversuche.

Für die prototypische Umsetzung im Rahmen des Projektes kam Emgu CV[33] – ein C#-Wrapper für die OpenCV[34]-Bibliothek – zum Einsatz. Bei OpenCV handelt es sich um eine freie Soft-

[32] http://www.rfid-webshop.com/product_info.php/info/p165_OMNIKEY---5513-Reader-Board-Compact-Mifare-Easy-Plug---Play.html
[33] http://www.emgu.com/wiki/index.php/Main_Page
[34] http://opencv.willowgarage.com/wiki/

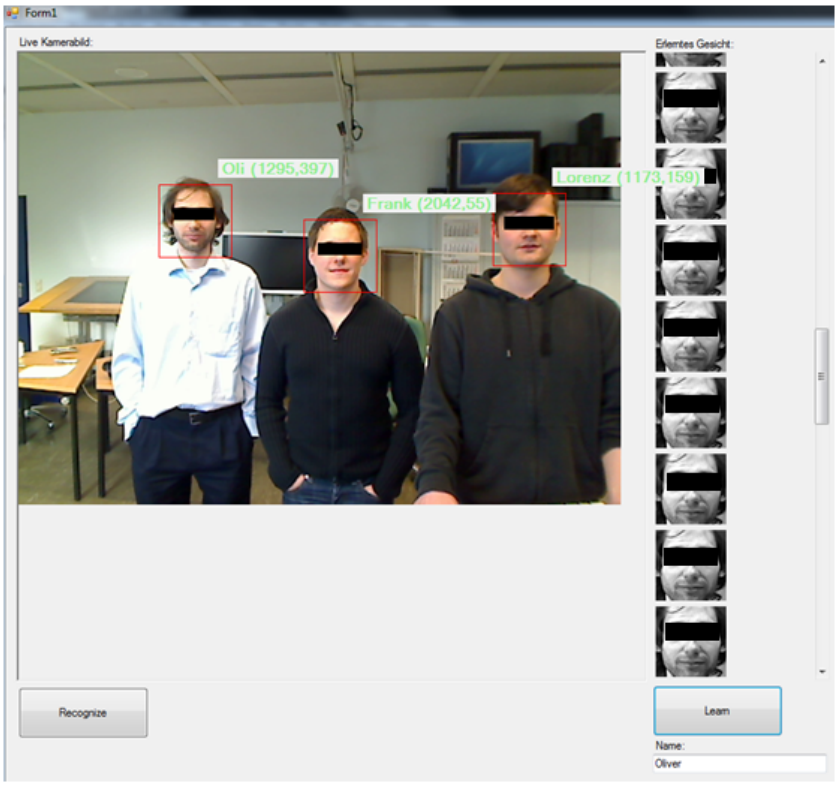

Abbildung 8: Bildschirmfoto der Testanwendung zur Gesichtserkennung

ware für Bildverarbeitung und maschinelles Sehen (Wikipedia (2011)). Zur Erfassung der Bilder wurde eine Logitech QuickCam Pro 9000 USB-Kamera verwendet.

In einem Vorverarbeitungsschritt müssen zunächst Gesichter innerhalb des Kamerabildes gefunden werden. Die Detektion erfolgte über so genannte *Haar-Kaskaden*, wie sie von Viola und Jones (2001) vorgestellt wurden, um bestimmte Muster aufzufinden. Mit OpenCV werden bereits Trainingsdaten ausgeliefert, so dass der Filter sofort zur Detektion von Gesichtern verwendet werden kann. In Abbildung 8 sind die erkannten Gesichter mit einem roten Rahmen markiert. Diese Erkennung ist jedoch nicht Rotationsinvariant, d. h. es werden prinzipiell nur Gesichter gefunden, bei denen die Kopfhaltung gerade ist.

Im nächsten Schritt werden Merkmale aus den ermittelten Bereichen extrahiert. Diese Merkmale werden während der Lernphase in einer Datenbank abgelegt, oder während der Erkennungsphase gegen die gelernten Merkmale verglichen, um das Gesicht zu identifizieren. Die Extraktion der Merkmale kann beispielsweise über die so genannten *Eigenfaces* nach Turk und Pentland (1991) erfolgen. Dieses Verfahren basiert auf der Hauptkomponentenanalyse, bei der eine ursprüngliche Menge von Merkmalen durch eine geeignete, kleinere Menge angenähert wird.

Abbildung 9 zeigt die Hauptkomponentenanalyse exemplarisch. Ausgehend von drei 2-dimen-

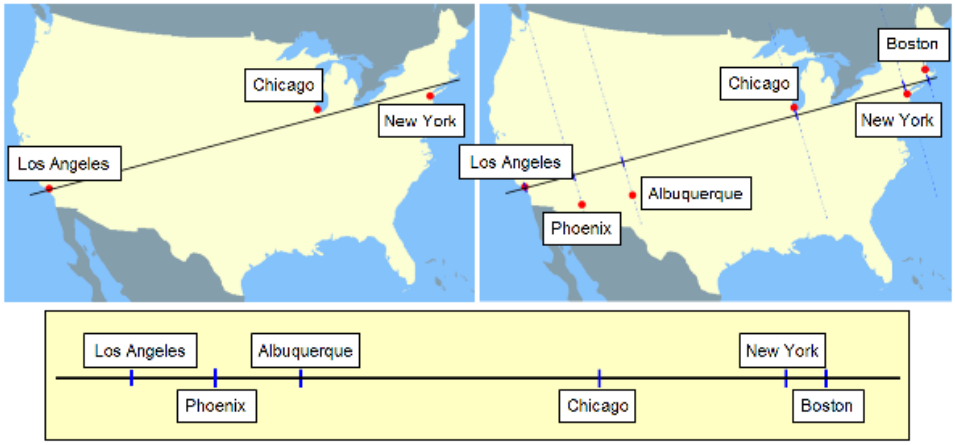

Abbildung 9: Beispiel für Hauptkomponentenanalyse aus Hewitt (2007)

sionalen Variablen (Los Angeles, Chicago und New York) wird eine Gerade gesucht, die diese Punkte am besten annähert. Damit lassen sich diese Städte auf eine Dimension reduzieren, indem der Punkt auf der Geraden bestimmt wird, der der entsprechenden Stadt am nächsten ist. Diese Vereinfachung führt aber dazu, dass Städte, die auf der selben Senkrechten zur o. g. Geraden stehen, auf den selben 1-dimensionalen Punkt abgebildet werden, obwohl sie ggf. räumlich sehr weit auseinander liegen.

Dieses Problem der Hauptkomponentenanalsyse zeigt sich besonders bei neuen Gesichtern, die noch nicht erlernt wurden. Die extrahierten und reduzierten Merkmalsvektoren werden mit der vorhandenen Datenbasis verglichen und es kann bestimmt werden, welcher erlernten Person dieser Vektor am nächsten kommt und wie groß die Distanz ist. Die Distanz im Subraum ist aber nicht intuitiv begreifbar und nicht zwingend aussagekräftig (vgl. Hewitt (2007)).

Zusammenfassend kann die verwendete Kombination aus Filtern zwar genutzt werden, um erlernte Gesichter zu erkennen, jedoch bietet sie (wie erwartet) eine eher geringe Sicherheit. Auch wenn diese Methode allein nicht zur einwandfreien Authentifizierung genutzt werden kann, ist eine Kombination mit anderen Techniken evtl. dennoch sinnvoll. Amazon[35] verwendet beispielsweise verschiedene Browser-Informationen, um den aktuellen Benutzer zu erraten und eine personalisierte Startseite anzubieten. Eine Authentifizierung erfolgt dennoch erst nach Eingabe von Benutzernamen und Passwort.

2.2.3 Smartphone

Schmidt u. a. (2010) beschreiben ein Verfahren, um Smartphones bei Berührungen eines Touchscreens zu identifizieren. Hierfür verwenden sie einen externen Beschleunigungssensor (130 Hz

[35] http://www.amazon.com/

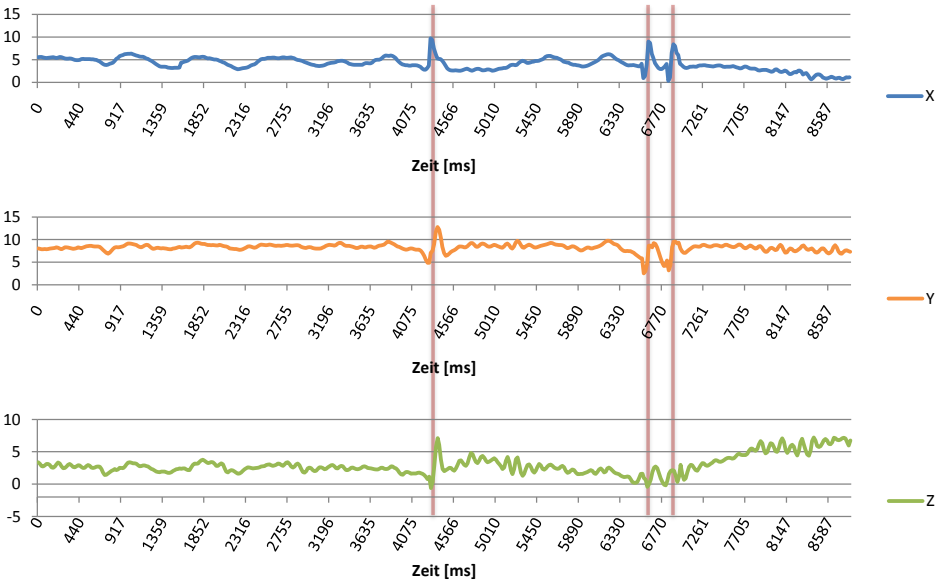

Abbildung 10: Rohdaten der Beschleunigungssensoren des HTC Desire

Datenrate), der per Bluetooth[36] mit einem Smartphone kommuniziert. Schmale, aber starke Ausschläge des Sensors werden als Kontakt des Smartphones mit dem Touchscreen interpretiert und an das Touchscreen-System übermittelt.

Touches, deren Berührungen kleiner als ein bestimmter Schwellwert sind, werden grundsätzlich als Berührung durch ein Smartphone interpretiert. Wenn eine solche Berührung zeitlich mit einer Kontaktmeldung eines Smartphones korreliert, kann das System dem Touch einen bestimmten Benutzer korrekt zuordnen.

In einem Experiment sollte dieses System mit den internen Beschleunigungssensoren eines HTC Desire nachgestellt werden. Abbildung 10 zeigt die erfassten Rohdaten der Sensoren. Es wurden drei Kontakte mit einer Tischoberfläche gemacht und in der Abbildung durch senkrechte, rote Linien gekennzeichnet. Obwohl die durchschnittliche Datenrate in diesem Test nur ca. 50 Hz betrug, lassen sich die Kontakte anhand der Erschütterungen erkennen.

Ein Kollision, also zwei quasi gleichzeitige Berührungen durch Smartphones, kann das System nicht auseinander halten. In den Untersuchungen von Schmidt u. a. (2010) haben sich bei drei gleichzeitig aktiven Benutzern nahezu keine Kollisionen ergeben. An dieser Stelle ist jedoch zu beachten, dass die Frequenz der Sensoren und der Touch-Erfassung hierbei eine wichtige Rolle spielen. Je geringer diese Frequenzen sind, desto höher ist die Wahrscheinlichkeit für Kollisionen.

[36] http://en.wikipedia.org/wiki/Bluetooth

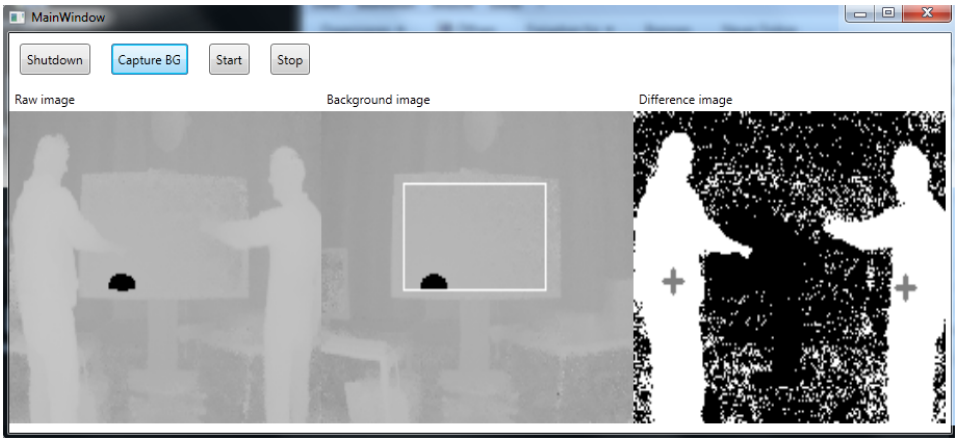

Abbildung 11: Testanwendung zum Benutzertracking mit TOF-Kamera

2.3 Benutzertracking

In Wilson (2010) wird eine Microsoft Kinect-Kamera[37] verwendet, um Touch-Eingaben zu erkennen. Der Vorteil dieser Technik ist, dass die Interaktionsfläche weder plan, noch mit Technik zur Berührungserkennung ausgestattet sein muss. Die geringe Genauigkeit gegenüber anderen Verfahren zur Umsetzung Multitouch-fähiger Oberflächen schränkt die Nützlichkeit jedoch ein.

Angelehnt an Wilson (2010) wurden im Rahmen des Projektes auch Versuche mit einer 3D-Kamera, einer sog. TOF[38]-Kamera[39], unternommen. Ziel der Versuche war es, Berührungen auf dem Touchscreen einem Benutzer zuordnen zu können. Abbildung 11 zeigt die entwickelte Testanwendung.

Zur Kalibrierung der Anwendung muss zunächst ein Tiefenbild des Hintergrunds aufgenommen werden, welches im mittleren Bereich der Abbildung zu sehen ist. Im laufenden Betrieb wird die Differenz des gespeicherten Hintergrundbildes und des aktuellen Kamerabildes (linker Bereich der Abbildung) erstellt und anhand eines Schwellwertes in ein Binärbild (rechter Bereich der Abbildung) überführt.

Im enstandenen Binärbild werden dann zusammenhängende Blöcke (so genannte Blobs) gesucht, die eine Mindestgröße überschreiten. Diese Blobs stellen die Benutzer dar und sind im so genannten Schwerpunkt des Blobs mit einem Kreuz markiert.

In der entwickelten Testanwendung kann im Hintergrundbild der interaktive Bereich des Touchscreens markiert werden. In der Abbildung ist dies durch den hellen Rahmen im mittleren Bereich gekennzeichnet. Meldet das Betriebssystem nun einen Touch auf der Oberfläche, so

[37] http://en.wikipedia.org/wiki/Kinect
[38] englisch.: time of flight
[39] http://en.wikipedia.org/wiki/Time-of-flight_camera

kann die entsprechende Bildschirmkoordinate in das Koordinatensystem der Kamera transformiert werden. Über eine Suche nach dem nächsten Blob lässt sich diese Berührung einem Benutzer zuordnen.

Dieses Verfahren ist jedoch eher für horizontale Touchscreens geeignet, da in der dargestellten Konfiguration die Berührungen sehr häufig durch den Körper des Benutzers verdeckt werden würde. Außerdem wurde das Informationssystem als Anwendung für einen einzelnen Benutzer konzipiert, weshalb eine Zuordnung von Touches zu Benutzern nicht erforderlich ist.

3 Zusammenfassung und Ausblick

3.1 Zusammenfassung

Diese Projektarbeitung beschreibt alle wesentlichen Aspekte, welche im Rahmen der Veranstaltung *Projekt 2* und für das hier beschriebenen Informationssystem umgesetzt wurden.

Es wurde einleitend die grundsätzliche Idee hinter dem Informationssystem beschrieben und auf vorangegangene Arbeiten im Masterstudium Bezug genommen. Neben den Systemkomponenten *Basisframework* und *Mitarbeiter und Professoren* wurde auch die umfangreichste Systemkomponente, die *Ideenplattform*, vorgestellt und auf dessen Konzept, Teilkomponenten und die Integration dieser Teilkomponenten eingegangen. Zusätzlich wurden verschiedene Authentifizierungsmodelle, wie RFID oder eine Gesichtserkennung, vorgestellt sowie auf das so genannte *Benutzertracking* eingegangen.

3.2 Ausblick

Wie bereits in Abschnitt 2.1 angedeutet, konnten nicht alle Bestandteile des Informationssystems im Rahmen der Veranstaltung *Projekt 2* umgesetzt werden. Im nächsten Schritt sollten die Systemkomponenten *Raumbelegung*, *Stockwerkansicht* und *Veranstaltungspläne* umgesetzt und in das Informationssystem integriert werden (vgl. Abschnitt 2.1.2).

Eine weitere Möglichkeit ergibt sich aus der Portierung des Informationssystems auf ein Smartphone: Das Windows Phone 7[40] unterstützt Silverlight[41], eine Web-basierte Sprache, welche Webseiten vor allem durch Animationen, Videos oder 3D-Effekte aufwerten soll (Wikipedia (2010a)). Da das Informationssystem auf WPF basiert (siehe Abschnitt 2.1.1) und Silverlight eine Untermenge der Funktionen von WPF anbietet, ist eine Portierung grundsätzlich möglich.

Auch ist es denkbar, das Informationssystem durch physikalische Elemente im näheren Umfeld zu erweitern. Bspw. nutzt Weiser einen einfachen von der Decke hängenden und mit einem Motor verbundenen Plastikfaden, dem so genannten *Dangling String*, um die Stärke von Netzwerktraffic zu visualieren (Weiser und Brown (1995)). Denkbar wäre es an dieser Stelle, durch solche visuellen Elemente bspw. die Länge von Mensaketten etc. zu veranschaulichen.

[40] http://www.microsoft.com/windowsphone/de-de/default.aspx
[41] http://www.silverlight.net/

A NHibernate Mappings

A.1 Basiskonfiguration

Listing 2: Basiskonfiguration für NHibernate

```
1  <?xml version="1.0" encoding="utf-8" ?>
2  <hibernate-configuration xmlns="urn:nhibernate-configuration
       -2.2">
3    <session-factory>
4      <property name="connection.provider">NHibernate.Connection
         .DriverConnectionProvider</property>
5      <property name="dialect">NHibernate.JetDriver.JetDialect,
         NHibernate.JetDriver</property>
6      <property name="connection.driver_class">NHibernate.
         JetDriver.JetDriver, NHibernate.JetDriver</property>
7      <property name="connection.connection_string">Provider=
         Microsoft.ACE.OLEDB.12.0;Data Source=C:/HAWMIRROR/test.
         accdb</property>
8      <property name="proxyfactory.factory_class">NHibernate.
         ByteCode.LinFu.ProxyFactoryFactory, NHibernate.ByteCode
         .LinFu</property>
9      <property name="show_sql">false</property>
10   </session-factory>
11 </hibernate-configuration>
```

A.2 Fachliches Modell

A.2.1 User-Mapping

Listing 3: NHibernate-Mapping des User-Modells

```
1  <?xml version="1.0" encoding="utf-8" ?>
2  <hibernate-mapping xmlns="urn:nhibernate-mapping-2.2"
3                     namespace="HibernateMsAccess.HawMirror"
4                     assembly="HibernateMsAccess">
5
6    <class name="User" table="Users">
7      <id name="ID" type="int">
8        <generator class="identity"></generator>
```

```
9     </id>
10
11    <property name="Username"  type="String"/>
12    <property name="Fullname"  type="String"/>
13  </class>
14 </hibernate-mapping>
```

A.2.2 Post-Mapping

Listing 4: NHibernate-Mapping des Post-Modells

```
1  <?xml version="1.0" encoding="utf-8" ?>
2  <hibernate-mapping xmlns="urn:nhibernate-mapping-2.2"
3                     namespace="HibernateMsAccess.HawMirror"
4                     assembly="HibernateMsAccess">
5
6    <class name="Post" table="Posts">
7      <id name="ID" type="int">
8        <generator class="identity"></generator>
9      </id>
10
11     <property name="AuthorID"  type="Int32"/>
12     <property name="Title"  type="String"/>
13     <property name="Body"  type="StringClob"/>
14     <property name="ParentID"  type="Int32"/>
15     <property name="URL"  type="String"/>
16   </class>
17 </hibernate-mapping>
```

A.2.3 Rating-Mapping

Listing 5: NHibernate-Mapping des Rating-Modells

```
1  <?xml version="1.0" encoding="utf-8" ?>
2  <hibernate-mapping xmlns="urn:nhibernate-mapping-2.2"
3                     namespace="HibernateMsAccess.HawMirror"
4                     assembly="HibernateMsAccess">
5
6    <class name="Rating" table="Ratings">
```

```
7     <id name="ID" type="int">
8       <generator class="identity"></generator>
9     </id>
10
11    <property name="PostID"  type="Int32"/>
12    <property name="AuthorID"  type="Int32"/>
13    <property name="Likes"  type="Int32"/>
14  </class>
15 </hibernate-mapping>
```

A.3 Facebook

A.3.1 User-Mapping

Listing 6: NHibernate-Mapping der Facebook-User

```
1  <?xml version="1.0" encoding="utf-8" ?>
2  <hibernate-mapping xmlns="urn:nhibernate-mapping-2.2"
3                     namespace="HibernateMsAccess.Facebook"
4                     assembly="HibernateMsAccess">
5
6    <class name="FBUser" table="FBUsers">
7      <id name="ID" type="String" />
8
9      <property name="UserID"  type="Int32"/>
10   </class>
11 </hibernate-mapping>
```

A.3.2 Post-Mapping

Listing 7: NHibernate-Mapping der Facebook-Posts

```
1  <?xml version="1.0" encoding="utf-8" ?>
2  <hibernate-mapping xmlns="urn:nhibernate-mapping-2.2"
3                     namespace="HibernateMsAccess.Facebook"
4                     assembly="HibernateMsAccess">
5
6    <class name="FBPost" table="FBPosts">
7      <id name="ID" type="int">
```

```
8       <generator class="identity"></generator>
9     </id>
10
11    <property name="FBPostID"  type="String"/>
12    <property name="ActorID"  type="String"/>
13    <property name="CreatedTime"  type="Int32"/>
14    <property name="UpdatedTime"  type="Int32"/>
15    <property name="Message"  type="StringClob"/>
16    <property name="PostID"  type="int"/>
17  </class>
18 </hibernate-mapping>
```

A.4 Twitter

A.4.1 Parameter-Mapping

Listing 8: NHibernate-Mapping der Twitter-Parameter

```
1  <?xml version="1.0" encoding="utf-8" ?>
2  <hibernate-mapping xmlns="urn:nhibernate-mapping-2.2"
3                     namespace="HibernateMsAccess.Twitter"
4                     assembly="HibernateMsAccess">
5
6    <class name="TLastPostId" table="TWLatestPostId">
7      <id name="ID" type="int">
8        <generator class="identity"></generator>
9      </id>
10     <property name="PostId"  type="int"/>
11   </class>
12 </hibernate-mapping>
```

A.4.2 User-Mapping

Listing 9: NHibernate-Mapping der Twitter-User

```
1  <?xml version="1.0" encoding="utf-8" ?>
2  <hibernate-mapping xmlns="urn:nhibernate-mapping-2.2"
3                     namespace="HibernateMsAccess.Twitter"
4                     assembly="HibernateMsAccess">
```

```
5
6   <class name="TUser" table="TWUsers">
7     <id name="ID" type="int">
8       <generator class="identity"></generator>
9     </id>
10    <property name="UserId"  type="Decimal"/>
11    <property name="TwitterName"  type="string"/>
12    <property name="TwitterDisplayName"  type="string"/>
13  </class>
14 </hibernate-mapping>
```

A.4.3 UserToUser-Mapping

Listing 10: NHibernate-Mapping der Twitter-User-Abbildung auf fachliche User

```
1  <?xml version="1.0" encoding="utf-8" ?>
2  <hibernate-mapping xmlns="urn:nhibernate-mapping-2.2"
3                     namespace="HibernateMsAccess.Twitter"
4                     assembly="HibernateMsAccess">
5
6    <class name="TUserToUser" table="TWUsersToUsers">
7      <id name="ID" type="int">
8        <generator class="identity"></generator>
9      </id>
10     <property name="TWUserId"  type="Decimal"/>
11     <property name="UserId"  type="int"/>
12   </class>
13 </hibernate-mapping>
```

A.4.4 Tweet-Mapping

Listing 11: NHibernate-Mapping der Twitter-Posts

```
1  <?xml version="1.0" encoding="utf-8" ?>
2  <hibernate-mapping xmlns="urn:nhibernate-mapping-2.2"
3                     namespace="HibernateMsAccess.Twitter"
4                     assembly="HibernateMsAccess">
5
6    <class name="Tweet" table="TWPosts">
```

```
7     <id name="ID" type="int">
8       <generator class="identity"></generator>
9     </id>
10    <property name="TweetId"  type="Decimal"/>
11    <property name="ReplyToTweetId"  type="Decimal"/>
12    <property name="TweetText"  type="String"/>
13    <property name="UserId"  type="Decimal"/>
14    <property name="CreatedDate"  type="System.DateTime"/>
15  </class>
16 </hibernate-mapping>
```

A.4.5 TweetToPost-Mapping

Listing 12: NHibernate-Mapping der Twitter-Post-Abbildung auf fachliche Posts

```
1  <?xml version="1.0" encoding="utf-8" ?>
2  <hibernate-mapping xmlns="urn:nhibernate-mapping-2.2"
3                     namespace="HibernateMsAccess.Twitter"
4                     assembly="HibernateMsAccess">
5
6    <class name="TweetToPost" table="TWPostsToPosts">
7      <id name="ID" type="int">
8        <generator class="identity"></generator>
9      </id>
10     <property name="TweetId"  type="Decimal"/>
11     <property name="PostId"  type="int"/>
12   </class>
13 </hibernate-mapping>
```

Literatur

[Alexa 2011] ALEXA: *Facebook.com*. Webseite. 2011. – URL http://www.alexa.com/siteinfo/facebook.com

[Barnkow 2010a] BARNKOW, Lorenz: *Eine Multitouch-fähige Küchentheke: Im Kontext des Living Place Hamburg*. Referat/Hausarbeit. 2010. – URL http://users.informatik.haw-hamburg.de/~ubicomp/projekte/master09-10-aw1/vortraege.html. – abgerufen am: 27.07.2010

[Barnkow 2010b] BARNKOW, Lorenz: *Eine Multitouch-fähige Küchentheke: Related Work*. Referat/Hausarbeit. 2010. – URL http://users.informatik.haw-hamburg.de/~ubicomp/projekte/master09-10-aw2/vortraege.html. – abgerufen am: 31.08.2010

[Blohm u. a. 2010] BLOHM, Ivo ; BRETSCHNEIDER, Ulrich ; LEIMEISTER, Jan M. ; KRCMAR, Helmut: *Does collaboration among participants lead to better ideas in IT-based idea competitions? An empirical investigation*. 2010. – URL http://www.uni-kassel.de/fb7/ibwl/leimeister/pub/JML_145.pdf. – abgerufen am: 09.02.2011

[Dourish und Bellotti 1992] DOURISH, Paul (Hrsg.) ; BELLOTTI, Victoria (Hrsg.): *Awareness and Coordination in Shared Workspaces*. Bd. *PDF*. Rank Xerox EuroPARC, 61 Regent St, Cambridge CB2 1AB UK : ACM Conference on Computer-Supported Cooperative Work, November 1992. – URL http://delivery.acm.org/10.1145/150000/143468/p107-dourish.pdf?key1=143468&key2=9027427921&coll=DL&dl=ACM&CFID=9326912&CFTOKEN=46738639

[Hewitt 2007] HEWITT, Robin: *Seeing With OpenCV, Part 4: Face Recognition With Eigenface*. Webseite. 2007. – URL http://www.cognotics.com/opencv/servo_2007_series/part_4/index.html. – abgerufen am: 09.02.2011

[InternetWorld 2010] INTERNETWORLD: *Statt fünf Sternen simples Daumen hoch*. Webseite. März 2010. – URL http://www.internetworld.de/Nachrichten/Medien/Medien-Portale/Youtube-Verbessertes-Bewertungssystem-Statt-fuenf-Sternen-simples-Daumen-hoch-26095.html

[Jain u. a. 2004] JAIN, A.K. ; ROSS, A. ; PRABHAKAR, S.: An introduction to biometric recognition. In: *Circuits and Systems for Video Technology, IEEE Transactions on* 14 (2004), Nr. 1, S. 4 – 20. – ISSN 1051-8215

[Koch und Möslein 2007] KOCH, Michael ; MÖSLEIN, Kathrin M.: *Die Rolle von Idea Mirrors zur Unterstützung von Innovation und Kooperation im Unternehmen*. 2007. – URL http://wi1.uni-erlangen.de/sites/wi1.uni-erlangen.de/files/Moeslein_WI_2007_Diskontinuierliche_Innovation_foerdern_0.pdf. – abgerufen am: 09.02.2011

[Koch und Toni 2004] KOCH, Michael (Hrsg.) ; TONI, Karlheinz (Hrsg.): *Community-Mirrors zur Unterstützung von Community-Treffen*. Bd. *PDF*. Gemeinschaften in Neuen Medien (GeNeMe), Oktober 2004. – URL http://sunschlichter0.informatik.tu-muenchen.de/lehrstuhl/personen/toni/pdf/toni2004a.pdf

[Ott 2010] OTT, Florian: *Einsatz großer Wandbildschirme zur Förderung diskontinuierlicher Innovation in der Softwarebranche*. Präsentation. Oktober 2010. – URL http://www.kooperationssysteme.de/wp-content/uploads/ott-2008-geneme-ideamirror-praesentation.pdf. – abgerufen am: 09.02.2011

[Philips 2004] PHILIPS: *mifare DESfire: Contactless Multi-Application IC with DES and 3DES Security MF3 IC D40*. Spezifikation. April 2004. – URL http://www.scdeveloper.com/datasheet/SFS075530.pdf

[Plötz 2008] PLÖTZ, Henryk: *Mifare Classic - Eine Analyse der Implementierung*. Diplomarbeit. Oktober 2008. – URL http://sar.informatik.hu-berlin.de/research/publications/SAR-PR-2008-21/SAR-PR-2008-21_.pdf

[Schlichter u. a. 1998] SCHLICHTER, Johann H. ; KOCH, Michael ; XU, Chengmao: Awareness - The Common Link Between Groupware and Community Support Systems. In: *Community Computing and Support Systems, Social Interaction in Networked Communities [the book is based on the Kyoto Meeting on Social Interaction and Communityware, held in Kyoto, Japan, in June 1998]*. London, UK : Springer-Verlag, 1998, S. 77–93. – URL http://portal.acm.org/citation.cfm?id=646698.701372. – ISBN 3-540-65475-5

[Schmidt u. a. 2010] SCHMIDT, Dominik ; CHEHIMI, Fadi ; RUKZIO, Enrico ; GELLERSEN, Hans: PhoneTouch: a technique for direct phone interaction on surfaces. In: *Proceedings of the 23nd annual ACM symposium on User interface software and technology*. New York, NY, USA : ACM, 2010 (UIST '10), S. 13–16. – URL http://doi.acm.org/10.1145/1866029.1866034. – ISBN 978-1-4503-0271-5

[Schwarzer 2010a] SCHWARZER, Jan: *Collaborative Programming mit Google Wave*. Referat/Hausarbeit. 2010. – URL http://users.informatik.haw-hamburg.de/~ubicomp/projekte/master09-10-aw1/vortraege.html. – abgerufen am: 21.02.2010

[Schwarzer 2010b] SCHWARZER, Jan: *Computer-Supported Cooperative Work (CSCW): Related Work*. Referat/Hausarbeit. 2010. – URL http://users.informatik.haw-hamburg.de/~ubicomp/projekte/master2010-aw2/schwarzer/bericht.pdf. – abgerufen am: 08.02.2011

[Sears 1991] SEARS, Andrew: Improving Touchscreen Keyboards: Design issues and a comparison with other devices. In: *Computers* 3 (1991), S. 253–269

[TAGnology 2008] TAGNOLOGY: *HF MIFARE Easy Module*. Manuel. November 2008. – URL http://www.rfid-webshop.com/shop/download/Reader/HF%2013.56%20MHz/ACG/ISO%2014443/TAGnology_UserManual_HF_MIFARE_Easy.pdf

[Turk und Pentland 1991] TURK, Matthew ; PENTLAND, Alex: Eigenfaces for recognition. In: *J. Cognitive Neuroscience* 3 (1991), January, S. 71–86. – URL http://portal.acm.org/citation.cfm?id=1326887.1326894. – ISSN 0898-929X

[Twitter.com 2011] TWITTER.COM: *Twitter Help Center - Häufig gestellte Fragen (FAQ)*. Webseite. 2011. – URL http://support.twitter.com/groups/31-twitter-basics/topics/104-welcome-to-twitter-support/articles/108034-h-xe4-ufig-gestellte-fragen-faq

[Viola und Jones 2001] VIOLA, P. ; JONES, M.: Rapid object detection using a boosted cascade of simple features. In: *Computer Vision and Pattern Recognition, 2001. CVPR 2001. Proceedings of the 2001 IEEE Computer Society Conference on* Bd. 1, 2001, S. I–511 – I–518 vol.1. – ISSN 1063-6919

[Weiser und Brown 1995] WEISER, Mark ; BROWN, John S.: *Designing Calm Technology*. Webseite. December 1995. – URL http://nano.xerox.com/hypertext/weiser/calmtech/calmtech.htm

[Wikipedia 2010a] WIKIPEDIA: *Microsoft Silverlight*. Webseite. Februar 2010. – URL http://de.wikipedia.org/wiki/Microsoft_Silverlight

[Wikipedia 2010b] WIKIPEDIA: *Mikroblogging*. Webseite. Dezember 2010. – URL http://de.wikipedia.org/wiki/Mikroblogging

[Wikipedia 2011a] WIKIPEDIA: *ISO/IEC 14443*. Webseite. Februar 2011. – URL http://de.wikipedia.org/wiki/ISO/IEC_14443

[Wikipedia 2011b] WIKIPEDIA: *Mifare*. Webseite. Februar 2011. – URL http://de.wikipedia.org/wiki/Mifare

[Wikipedia 2011] WIKIPEDIA: *OpenCV*. Webseite. 2011. – URL http://en.wikipedia.org/wiki/OpenCV. – abgerufen am: 09.02.2011

[Wikipedia 2011a] WIKIPEDIA: *RFID*. Webseite. Februar 2011. – URL http://de.wikipedia.org/wiki/RFID

[Wikipedia 2011b] WIKIPEDIA: *Twitter*. Webseite. Februar 2011. – URL http://de.wikipedia.org/wiki/Twitter

[Wilson 2010] WILSON, Andrew D.: Using a depth camera as a touch sensor. In: *ACM International Conference on Interactive Tabletops and Surfaces*. New York, NY, USA : ACM, 2010 (ITS '10), S. 69–72. – URL http://doi.acm.org/10.1145/1936652.1936665. – ISBN 978-1-4503-0399-6