Hochschule für Angewandte Wissenschaften Hamburg
Hamburg University of Applied Sciences

Studienarbeit

Entwicklung eines Systems zur Clusteranalyse von Benutzerprofilen

vorgelegt von
Christian Butrynowski
am 8. Februar 2005

Studiengang Softwaretechnik

Betreuer: Prof. Dr. Kai von Luck

Fachbereich Elektrotechnik und Informatik
Department of Electrical Engineering and Computer Science

Inhaltsverzeichnis

1 Einleitung

1.1 Motivation

Computer spielen seit Jahren eine zunehmend größere Rolle in unserem Alltag. Mittlerweile besitzen viele Menschen nicht bloß einen PC, sondern viele weitere Computer - oftmals ohne sich dessen bewußt zu sein. Dazu gehören Mobiltelefone, die mit jeder Generation weitere Funktionen bieten welche weit über das Telefonieren hinaus gehen. Dazu zählen digitale Videorecorder, Navigationssysteme in Fahrzeugen, MP3-Player oder auch PDAs[1].

Überall verfügbare Computer wie Mobiltelefone und PDAs bieten die Möglichkeit für eine ungeahnte Anzahl von Anwendungen. So ist ein mobiler Reiseführer vorstellbar, der dem Touristen je nach Standort Informationen zum gerade besichtigten Bauwerk anzeigt. Oder die vom Fraunhofer-Institut entwickelte PDA-Anwendung die den Fußballfan bei der WM 2006 vor, während und nach dem Spiel mit zusätzlichen Informationen versorgen soll(MAV-Online).

In dieser Arbeit soll eine mögliche Anwendung für den Einsatz eines PDAs in einem Urlaubsclub entwickelt werden. Die Anwendung soll dazu dienen Menschen mit ähnlichen Interessen leichter zusammenzubringen und ihren Urlaub damit zu verschönern.

1.2 Gliederung

Im Kapitel 2 soll ein mögliches **Szenario** für die Anwendung gezeichnet werden, um dem Leser eine bildhafte Vorstellung zu ermöglichen. In Abschnitt 3 wird die Aufgabenstellung **analysiert**. Es werden die Rollen und Anwendungsfälle vorgestellt, und es werden algorithmische Lösungsansätze gezeigt. Das folgende Kapitel 4 befasst sich mit dem **Design** und der **Implementierung** der Anwendung. Dabei wird zunächst die grobe Systemarchitektur vorgestellt um schließlich in die Tiefe zu gehen und am Ende das Klassenmodell vorzustellen. Die Gestaltung und die Benutzung der Anwendung ist Thema von Abschnitt 5, in dem die Anwendung anhand einer konkreten **Fallstudie** vorgestellt wird. Zu guter Letzt gibt das

[1] Personal Digital Assistant

Kapitel 6 einen **Ausblick** auf unbearbeitete Aspekte von get2gether, die in einer endgültigen Version Beachtung finden müßten.

2 Szenario

Das folgende Szenario soll veranschaulichen, was get2gether leisten soll. Auch wenn im Rahmen dieser Arbeit nicht alle Aspekte vollständig umgesetzt werden, soll das Szenario als Vision einer möglichen Fortentwicklung des Projektes dienen.

2.1 Kalle im Urlaubsclub

Single Kalle möchte in Urlaub fahren und dabei gerne Leute kennen lernen mit denen er seine liebsten Freizeitaktivitäten teilen kann. Da entdeckt er in einer Broschüre des Urlausclubs get2gether folgenden Satz: “Urlaub unter “seinesgleichen“: unkompliziert, ausgelassen und relaxt in einer Gruppe von Gleichgesinnten. . . “. Kurzentschlossen nimmt er das Angebot wahr und erreicht wenig später auch schon den modernen Urlaubsclub get2gether. Als Kalle im Hotel ankommt, erhält er an der Rezeption neben seinem Zimmerschlüssel auch einen PDA. Die Dame bittet ihn dabei freundlich, zwecks optimaler Urlaubsgestaltung, doch mal einen Blick auf die Software get2gether zu riskieren.

Neugierig und hoffnungsfroh lässt sich Kalle nicht zweimal bitten. Das Programm fragt ihn nach einigen persönlichen Daten wie Geschlecht, Alter und ob er Single sei. Anschließend wird er aufgefordert einige potentielle Interessengebiete auf einer Skala von 1 bis 10 einzuordnen. Die Interessengebiete umfassen dabei u. a. allerlei Sport- und Freizeitaktivitäten, die im Club angeboten werden (Golfen, Reiten, Segeln, Surfen, Tanzen. . .). Nachdem er alle Eingaben getätigt hat, erhält Kalle die Rückmeldung, dass seine Daten übertragen und gespeichert werden. Kurz darauf macht ihm get2gether den Vorschlag sich beim Abendessen doch mal an Tisch 7 zu setzen. Dort säßen weitere Personen die ähnlichen Interessen wie er nachgingen. Nun möchte Kalle die weiteren Funktionalitäten ausprobieren und gibt an, dass er gerne morgen um 14.00 eine Partie Tennis spielen möchte, und ob ihm der Assistent nicht einen Spielpartner aussuchen könnte. Kurze Zeit später erscheint auf dem PDA des tennisbegeisterten Klaus die Frage, ob er nicht Lust hätte morgen um 14.00 eine Partie Tennis gegen den ebenso tennisbegeisterten Kalle zu spielen. Klaus ist davon sehr angetan, antwortet natürlich mit Ja, woraufhin Kalle eine Bestätigung erhält.

3 Analyse

In diesem Abschnitt sollen zum Einen die Rollen beschrieben werden, die die Benutzer des Systems annehmen können. Zum Anderen wird anhand von Anwendungsfällen (“Use Cases“) aufgezeigt, wie die Benutzer als Spieler ihrer Rollen, mit dem System interagieren.

Anschließend sollen gewisse Aspekte bezüglich der Nähe und Distanz von Profilen zueinander betrachtet werden. Dabei werden zwei Möglichkeiten der Messung von Distanzen zwischen Profilen dargestellt. Schließlich wird gezeigt welche Algorithmen eingesetzt werden können um die gestellte Aufgabe zu lösen.

3.1 Rollen

Die Anwendung unterscheidet zwischen den zwei Rollen Administrator und Feriengast, die jeweils unterschiedliche Handlungen vollziehen können.

Administrator

Als Administrator soll der Benutzer auf Seiten des Urlaubsclubs bezeichnet werden. Er hat also nichts mit einem System-Administrator gemein. In dieser Rolle können diverse Angestellte des Urlaubsclubs vereinigt sein. So könnten der Rezeptionist, der Animateur oder auch der Hotel-Manager in die Lage gelangen die Anwendung in der Rolle des Administrators zu bedienen.

Feriengast

Der Feriengast steht für den Touristen der eine Reise in den Urlaubsclub gebucht und angetreten hat - z. B. Kalle aus dem Szenario 2.1. Der Feriengast soll die Anwendung über ein Terminal bedienen. Dies entspricht zwar nicht dem Bild aus Szenario 2.1 - wo der Feriengast einen PDA bedient - soll aber zur Vereinfachung in dieser Form umgesetzt werden.

3.2 Use Cases

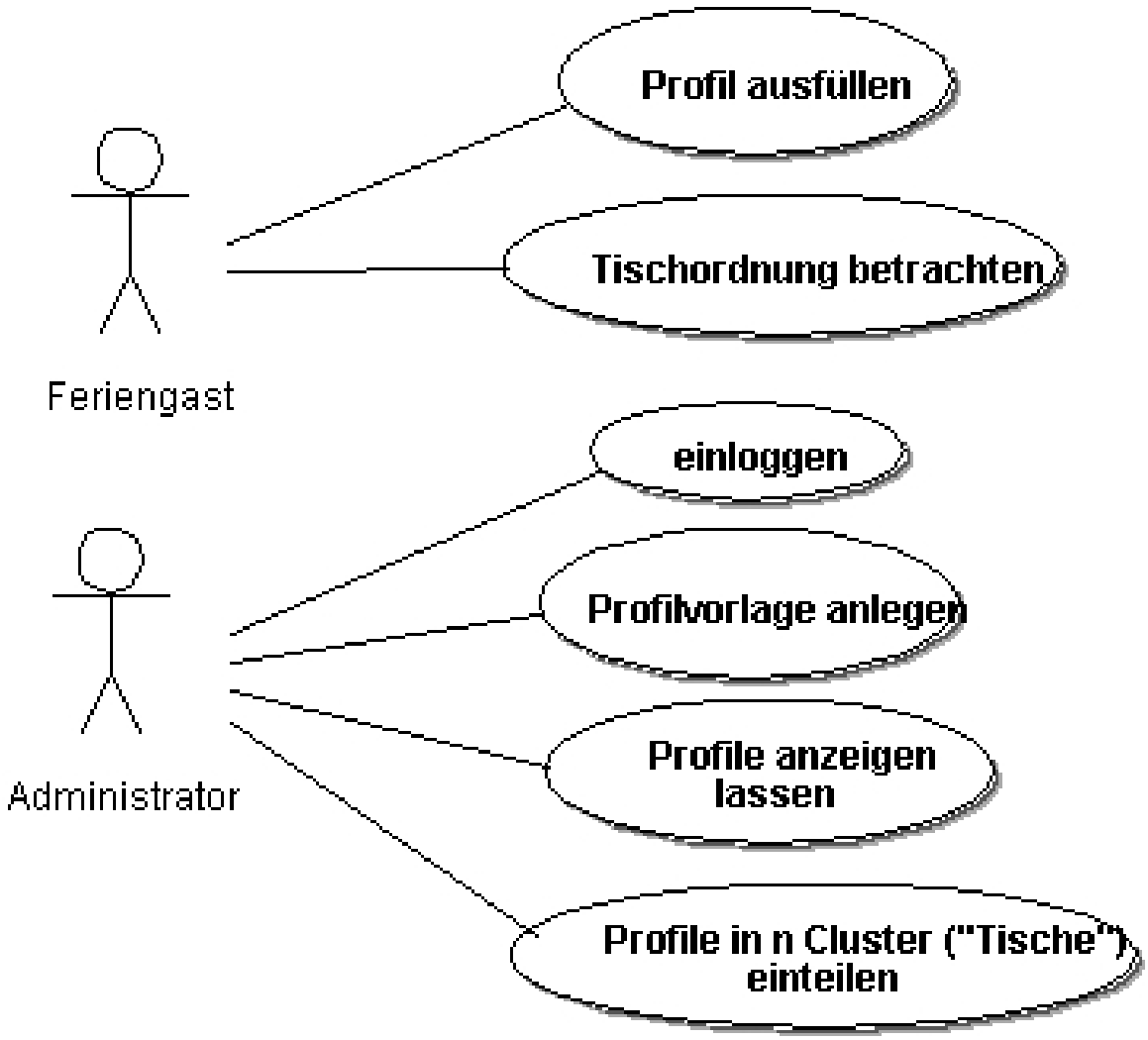


Abbildung 3.1: Die Anwendungsfälle der Rollen Feriengast und Administrator

Administrator

Einloggen

Der Administrator muß sich mit seinem Benutzernamen und einem Passwort in die Anwendung einloggen. Erst dann kann er die verfügbaren Dienste nutzen. Dies dient der Sicherheit und dem Mißbrauch der Anwendung durch Feriengäste oder durch Dritte.

Profilvorlage anlegen

Bevor die Feriengäste ihre Profile ausfüllen können, muß der Administrator eine Profilvorlage erstellen. Die Profilvorlage definiert eine bestimmte Art von Profilen. Es sind jeweils beliebig viele Attribute anzugeben (z. B. Tauchen, Tanzen. . .) und zu jedem Attribut ein zugehöriges Intervall (z. B. 1 bis 10). Dies bedeutet, dass ein Feriengast beim Ausfüllen seines Profils beim Attribut Tauchen Werte zwischen 1 (gar kein Interesse) und 10 (lebensnotwendig) eingeben kann. Die Intervalle können für jedes Attribut einzeln vergeben werden. Folgende Tabelle soll eine mögliche Profilvorlage darstellen:

Attribut	untere Grenze	obere Grenze
Tauchen	1	10
Schwimmen	1	5
Tanzen	1	3
Tennis	1	10
Wandern	1	10

Tabelle 3.1: Beispiel einer Profilvorlage

Profile anzeigen lassen

Mit Hilfe dieser Funktionalität kann sich der Administrator alle bereits eingegebenen Profile nebst Namen der zugehörigen Personen anzeigen lassen. Es werden sowohl automatisch erzeugte als auch manuell eingegebene Profile als Tabelle angezeigt. Tabelle 3.2 zeigt ein mögliches Ergebnis dieser Funktion, gemäß der weiter oben beschriebenen Profilvorlage.

Name	Tauchen	Schwimmen	Tanzen	Tennis	Wandern
Hans Meiser	3	3	1	8	5
Rudi Carrell	5	4	3	2	7
Kai Pflaume	10	5	1	6	2

Tabelle 3.2: Beispiel eine Profilliste

Profile in n Cluster (“Tische“) einteilen

Hierin besteht die Hauptfunktionalität von get2gether. Die eingegebenen Profile sollen auf Sitzplätze verteilt werden. Das Ziel besteht darin, dass beim Abendessen möglichst jeweils Personen mit ähnlichen Interessen an einem Tisch sitzen, um sich besser kennenlernen und austauschen zu können. Der Administrator kann also eine bestimmte Anzahl von Tischen

vorgeben. Anschließend werden alle Personen - bzw. ihre gesammelten Profile - auf die Anzahl der Tische verteilt.

Schließlich erhält der Benutzer eine Übersicht der Tischeinteilung. Hier ist nun auch eine Auflistung der Profile zu finden, damit sich der Administrator überzeugen kann, dass die Personen an jedem Tisch tatsächlich ähnliche Interessen haben.

Feriengast

Profil ausfüllen

Der Feriengast geht zu einem der im Club aufgestellten Terminals. Dort gibt er seinen Namen ein und füllt das Profil aus, indem er zu jedem Attribut einen Wert angibt der seine Präferenz dafür widerspiegelt. Vorher muß natürlich von einem Administrator eine Profilvorlage eingegeben worden sein.

Tischordnung betrachten

Natürlich möchte der Feriengast zu guter Letzt wissen, an welchem Tisch er beim Abendessen sitzen soll. Hierfür kann er eine Übersicht über die Tischordnung einsehen. Vorher müssen die Profile natürlich von einem Administrator eingeteilt worden sein.

Um keine Datenschutzrichtlinien zu verletzen, sollte dieser Übersicht nicht zu entnehmen sein, welche Werte die einzelnen Profile beinhalten.

3.3 Nähe und Distanz

Um die gespeicherten Profile in Gruppen bzw. Tische einteilen zu können, ist es zunächst notwendig zu definieren wann zwei Profile einander ähnlich sind und wie ähnlich sie sich sind. Es wird also ein Maß benötigt, welches z. B. zunehmend niedriger wird, je ähnlicher sich zwei Profile sind und es wird eine Berechnungsvorschrift benötigt um das jeweilige Maß zu bestimmen.

Im Folgenden werden zwei Ansätze vorgestellt, die im Rahmen dieser Arbeit untersucht wurden.

Euklidische Distanz

Die Euklidische Distanz ist eine einfache Möglichkeit um ein Maß für die Ähnlichkeit zweier Profile zu ermitteln. Dabei wird jedes Profil als Punkt im n-dimensionalen Raum gesehen, dessen Koordinaten durch einen Vektor angegeben werden. Die Distanz zwischen zwei Vektoren wird als Länge (“Euklidische Norm“ (Net-Lexikon)) des Differenzvektors der beiden Vektoren bestimmt.

Die Euklidische Distanz kann nach folgender Formel berechnet werden (siehe: (Werner, 2004, S. 48)):

$$|a - b| = \sqrt{\sum_{j=1}^{n} (b_j - a_j)^2} \tag{3.1}$$

Zur besseren Veranschaulichung betrachten wir erneut die Beispiel-Profile aus 3.2. Um die Profile besser darstellen zu können, nehmen wir zudem an, die Profilvorlage bestünde nur aus Tauchen und Schwimmen, so dass wir uns im zweidimensionalen Raum befinden. Wie sehen nun die Maße für die Distanzen der Profile unserer Herren Showmaster aus?

Die folgende Abbildung zeigt ein zweidimensionales Koordinatensystem in dem die Punkte *Meiser*, *Carrell* und *Pflaume* für die Merkmale **Tauchen** und **Schwimmen** eingetragen sind:

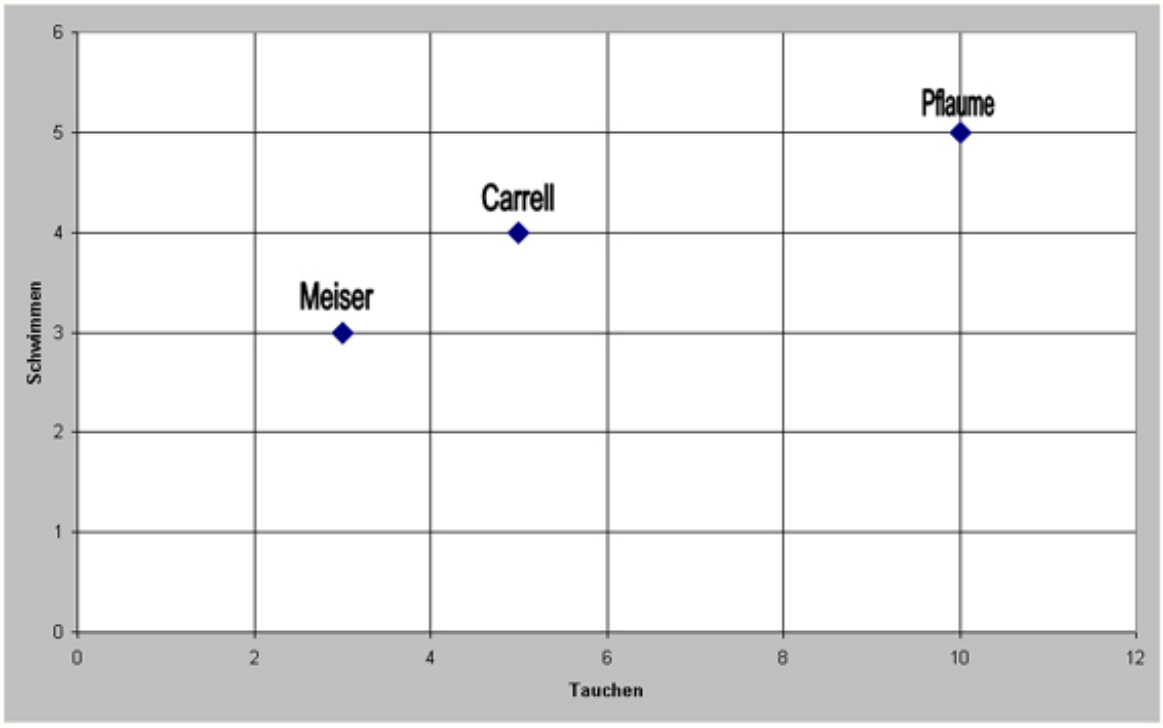


Abbildung 3.2: Die Profile der Showmaster als Punkte im Koordinatensystem

Es scheint als lägen die Herren Meiser und Carrell recht nah beieinander, wogegen die Herren Meiser und Pflaume einen eher hohen Abstand zueinander aufweisen. Dies soll per Rechnung gezeigt werden:

$$\begin{aligned} |Meiser - Carrell| &= \sqrt{(5-3)^2 + (4-3)^2} \\ &= \sqrt{2^2 + 1^2} \\ &= \sqrt{5} \\ &= 2,24 \end{aligned}$$

$$\begin{aligned} |Meiser - Pflaume| &= \sqrt{(10-3)^2 + (5-3)^2} \\ &= \sqrt{7^2 + 2^2} \\ &= \sqrt{53} \\ &= 7,28 \end{aligned}$$

$$\begin{aligned} |Pflaume - Carrell| &= \sqrt{(5-10)^2 + (4-5)^2} \\ &= \sqrt{-5^2 + (-1)^2} \\ &= \sqrt{26} \\ &= 5,10 \end{aligned}$$

Legt man also die Euklidische Distanz als Maß der Ähnlichkeit zugrunde, ist die Ähnlichkeit zwischen den Herren Meiser und Carrell am Höchsten, während die Interessen der Herren Meiser und Pflaume schon deutlich divergieren.

Gewichtete Euklidische Distanz

Nun wäre es vorstellbar, dass in gewissen Anwendungsfällen bestimmte Attribute von Profilen eine größere Rolle spielen sollten als andere.

Eine einfache Möglichkeit dafür ist die Anpassung der euklidischen Distanz. Es kann jedem Attribut ein Gewicht zwischen Null und Eins vergeben werden. Die Summe der Gewichte sollte Eins betragen. Nun kann die Formel zur Berechnung der euklidischen Distanz entsprechend modifiziert werden:

$$|a - b| = \sqrt{\sum_{j=1}^{n} w_j * (b_j - a_j)^2} \tag{3.2}$$

Hierbei entspricht w_j dem Gewicht für das Attribut mit dem Index j.

Nun können wir wieder das Beispiel von oben aufgreifen. Wir vergeben Gewichte für die Attribute Tauchen und Schwimmen und zwar wie folgt:
$w_t = 0,2$
$w_s = 0,8$

Mit diesen Gewichten erhalten wir nun folgende Distanzwerte:

$$\begin{aligned} |Meiser - Carrell| &= \sqrt{0,2 * (5-3)^2 + 0,8 * (4-3)^2} \\ &= \sqrt{0,2 * 2^2 + 0,8 * 1^2} \\ &= \sqrt{0,2 * 4 + 0,8 * 1} \\ &= \sqrt{1,6} \\ &= 1,26 \end{aligned}$$

$$\begin{aligned} |Meiser - Pflaume| &= \sqrt{0,2 * (10-3)^2 + 0,8 * (5-3)^2} \\ &= \sqrt{0,2 * 7^2 + 0,8 * 2^2} \\ &= \sqrt{0,2 * 49 + 0,8 * 4} \\ &= \sqrt{13} \\ &= 3,61 \end{aligned}$$

$$\begin{aligned} |Pflaume - Carrell| &= \sqrt{0,2 * (5-10)^2 + 0,8 * (4-5)^2} \\ &= \sqrt{0,2 * (-5)^2 + 0,8 * (-1)^2} \\ &= \sqrt{0,2 * 25 + 0,8 * 1} \\ &= \sqrt{5,8} \\ &= 2,41 \end{aligned}$$

Es ist zu beobachten, dass alle drei Distanzen deutlich niedriger geworden sind. So spielt die eigentlich große Distanz von Fünf zwischen den Herren Pflaume und Carrell beim Attribut Tauchen durch die geringe Gewichtung eine viel kleinere Rolle als vorher. Dagegen wird die geringe Distanz von Eins beim Attribut Schwimmen nun durch die hohe Gewichtung deutlicher hervorgehoben, so dass die Gesamtdistanz dieser beiden Profile nun weniger als die Hälfte der ungewichteteten euklidischen Distanz beträgt.

Fuzzy-Logik

Die Fuzzy-Logik[1] wurde 1965 von L.A. Zadeh an der Universität von Berkeley, USA entwickelt und vorgestellt (Zadeh, 1965). Die Fuzzy-Logik behandelt unscharfe Aussagen, wie sie im Sprachgebrauch permanent benutzt werden. Dies sind Aussagen, denen kein eindeutiger Wahrheitswert zugeordnet werden kann. Die Aussage "*Hamburg liegt an der Elbe*" ist eine eindeutige Aussage, der ein Wahrheitswert von Eins zugeordnet werden kann. Wohingegen es sich bei der Aussage "*Hamburg ist eine große Stadt*" um eine unscharfe Formulierung handelt. So könnte ein Bewohner Seouls - das rund 12 Mio. Einwohner (Worldatlas) zählt - Hamburg mit seinen ca. 1,7 Mio. Einwohnern (Citypopulation) durchaus als kleine Stadt ansehen.

Um mit derartigen unscharfen Aussagen umgehen zu können, wurden in der Fuzzy-Logik die **Fuzzy-Mengen** eingeführt. Dabei beschreiben die sog. **charakteristischen Funktionen** (bzw. **Zugehörigkeitsfunktionen**) die Zugehörigkeit zu einer Klasse (Lämmel und Cleve, 2001, S. 87).

Die charakteristische Funktion μ_F ordnet den Elementen einer Menge X einen Wert zwischen Null und eins zu, der den Zugehörigkeitsgrad r der Elemente zu einem unscharfen Begriff darstellt - analog zu (Babic, 2003, S. 79f).

Um im Bild der großen Städte zu bleiben, definieren wir eine Funktion welche Städte mit einer Einwohnerzahl unter 250.000 der Menge der *nicht großen* Städte zuordnet, während Städte mit mehr als 1 Mio. Einwohnern der Menge der *großen* Städte zugeordnet werden. Allen Städten deren Einwohnerzahl dazwischen liegt, wird mit Hilfe der Zugehörigkeitsfunktion der Grad r zum Begriff *groß* zugewiesen:

$$\mu_F(x) = \begin{cases} 0 & \text{für} \quad 0 \leq x \leq 250000 \\ \frac{4}{3000000}x - 333333,\overline{3} & \text{für} \quad 250000 \leq x \leq 1000000 \\ 1 & \text{für} \quad x > 1000000 \end{cases} \tag{3.3}$$

Zur besseren Veranschaulichung ist in Abb. 3.3 der Graph der Funktion dargestellt.

Laut dieser Funktion ist Frankfurt am Main mit etwa 640.000 Einwohnern mit einer Zugehörigkeit von 0,52 eine große Stadt.

Es lassen sich auf Fuzzy-Mengen u. a. die gleichen Mengenoperationen (Durchschnitt, Vereinigung usw.) anwenden wie auf Mengen der klassischen Mengenlehre. Hier sei aber auf die einschlägige Literatur wie (Görz u. a., 2003, S. 302ff) oder (Lämmel und Cleve, 2001, S. 87) verwiesen.

[1] Engl.: fuzzy = unscharf, verschwommen

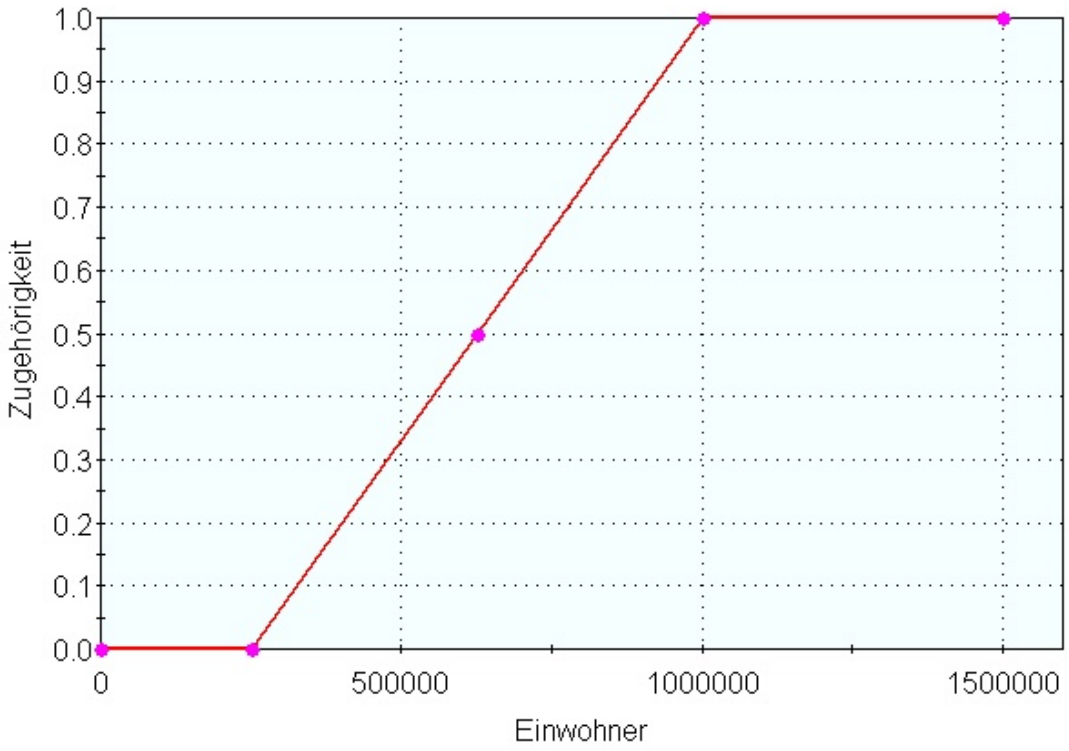


Abbildung 3.3: Die Zugehörigeitsfunktion für große bzw. nicht große Städte

Mit Hilfe der Fuzzy-Logik ist nun eine raffiniertere Berechnung der Nähe bzw. Distanz zweier Profile zueinander denkbar, als es die Euklidische Distanz ermöglicht.

So wäre eine Zugehörigkeitsfunktion denkbar, welche für alle Merkmale eines Profils die Nähe der Werte definiert. Die Funktion könnte für jedes Merkmal gesondert betrachtet werden, so dass z. B. bei manchen Merkmalen die gleiche Distanz mehr wiegt als bei anderen Merkmalen.

Des Weiteren könnte je nach Anwendungsfall die Zugehörigkeitsfunktion für das Merkmal Geschlecht variieren. Im Urlaubsclub für Singles könnte die Zugehörigkeitsfunktion beim Vergleich eines Profils eines Mannes mit dem Profil einer Frau eine Eins ergeben. Wohingegen die Funktion in einem Urlaubsclub für Burschenschafter[2] wohl eher bei zwei männlichen Profilen eine Zugehörigkeit von Eins ergeben würde.

[2] Die Idee ist zugegebenermaßen recht abwegig

3.4 Optimierungsproblem

Bei dem Problem der Unterteilung der Profile in Gruppen (bzw. Tische) handelt es sich offenbar um ein Optimierungsproblem, denn es gilt aus einer Menge vieler möglicher Lösungen ("Tischbesetzungen") die bestmögliche oder zumindest eine zufriedenstellende zu finden. Das Problem läßt sich nun aus - mindestens - zwei Blickwinkeln betrachten. Zum einen kann es als sog. *Reihenfolge-* und zum anderen als *Clusteringproblem* angesehen werden. Beides wird im folgenden näher erläutert.

Reihenfolgeproblem

Bei dem bekanntesten Reihenfolgeproblem handelt es sich um das *Travelling Salesman Problem (TSP)*. Hierbei gilt es für einen Handlungsreisenden die kürzeste Route durch n Städte zu finden, wobei jede Stadt nur einmal besucht werden darf und der Reisende am Ende wieder in seiner Ausgangsstadt ankommen soll.

Das Problem der Verteilung der Personen auf die Tische lässt sich in ähnlicher Form formulieren. Die einzelnen Profile können als n Städte betrachtet werden. Nun will ein Reisender m Reisen durch diese Städte unternehmen. Er will bei jeder Reise nur neue Städte besuchen, die er auf den vorherigen Reisen noch nicht gesehen hat. Es gilt nun für jede einzelne Reise die Strecke zu minimieren - d. h. die "Entfernung" der am gleichen Tisch sitzenden Personen wird minimiert.

Hopfield Netze

Es gibt zahlreiche Ansätze zur Lösung des TSP. An dieser Stelle soll eine mögliche Lösung auf Basis von Hopfield-Netzen gezeigt werden. Anschließend soll skizziert werden, wie das "Tischproblem" mit Hilfe von Hopfield-Netzen gelöst werden könnte.

Bei Hopfield-Netzen handelt es sich um sog. **autoassoziative Netze**, welche streng genommen keine neuronalen Netze sind. Alle Neuronen sind untereinander verbunden, und es existiert keine Ein- und keine Ausgabeschicht. Eine Eingabe wird durch initiale Aktivierungen der Neuronen erreicht. Daraufhin arbeitet das Netz so lange, bis keine Änderung mehr zwischen Zustand $n-1$ und Zustand n stattfindet. Der Aktivierungszustand n stellt die Ausgabe des Netzes dar. Hopfield-Netze sollen an dieser Stelle nicht im Detail erläutert werden. Der interessierte Leser kann alles Weitere z. B. in (Lämmel und Cleve, 2001, S. 239ff), (Zell, 2003, S. 197ff) oder auch bei (Callan, 2003, S. 102ff) nachschlagen.

Hopfield-Netze beinhalten eine **Energiefunktion** E welche sich bei jedem Arbeitsschritt verringert. E muß ein Minimum erreichen, sobald sich das Netz im Zustand n befindet.

Nun kann die Energiefunktion auf das jeweils zu lösende Problem angepaßt werden. Will man das TSP mit einem Hopfield-Netz modellieren, so ließe sich die Strecke Hamburg, Berlin, München, Köln und Bremen in dieser Reihenfolge durch ein Netz mit folgender Aktivierung darstellen (eine Eins kennzeichnet jeweils ein aktiviertes Neuron):

Stadt/Reihenfolge	1	2	3	4	5
Hamburg	1	0	0	0	0
Bremen	0	0	0	0	1
Köln	0	0	0	1	0
München	0	0	1	0	0
Berlin	0	1	0	0	0

Tabelle 3.3: Matrix der Strecke Hamburg, Berlin, München, Köln, Bremen

Die Energiefunktion E die mit Hilfe dieses Netzes minimiert werden soll, muß folgende Bedingungen erfüllen (Zell, 2003, S. 203)

1. E darf nur minimal sein für die Lösungen, die genau eine Eins in jeder Zeile und Spalte haben.
2. E muß für Lösungen mit kürzerer Pfadlänge geringer sein als für solche mit längerer Pfadlänge.

Um diese Bedingungen zu erfüllen, kann eine Funktion definiert werden die aus vier Termen besteht. Die genaue mathematische Formulierung der Terme kann bei Interesse in (Zell, 2003, S. 203) nachgeschlagen werden. Hier soll nur auf die Semantik der Terme eingegangen werde:

- der erste Term darf nur dann Null ergeben, wenn jede Zeile eine Eins enthält. Damit wird sichergestellt, dass jede Stadt nur genau einmal besucht wird.
- der zweite Term darf nur Null ergeben, wenn in jeder Spalte genau eine Eins vorkommt. Dies stellt sicher, dass nicht mehrere Städte gleichzeitig besucht werden können.
- der dritte Term darf nur dann Null ergeben, wenn genau n (mit n = Anzahl der Städte) Einsen in der Matrix vorkommen. Dies stellt sicher, dass auch wirklich jede Stadt besucht wird.
- der vierte Term gibt die Länge der aktivierten Strecke an.

Auf ähnliche Weise, läßt sich nun das “Tischproblem“ mit Hilfe eines Hopfield Netzes modellieren. Der Benutzer müßte vorgeben, auf wie viele Tische die Profile zu verteilen sind. Die mögliche Matrix für drei Tische ist in Tabelle 3.4 dargestellt.

Auch für diese Problemstellung läßt sich nun eine Energiefunktion E' definieren, die aus mehrerern Termen besteht:

Profil/Tischnummer	1	2	3
Kai Pflaume	1	0	0
Rudi Carrell	0	0	1
Hans Meiser	0	0	1
Thomas Gottschalk	0	1	0
Günther Jauch	1	0	0
Ulrich Meyer	0	1	0
Frank Elstner	0	0	1
Peter Bond	1	0	0

Tabelle 3.4: Matrix der Verteilung von Showmastern auf Tische

- der erste Term darf nur dann Null ergeben, wenn jede Zeile eine 1 enthält. Damit wird sichergestellt, dass jede Person an genau einem Tisch sitzt.
- der zweite Term darf nur Null ergeben, wenn in jeder Spalte mindestens zwei Einsen vorkommen. Dies stellt sicher, dass an jedem Tisch mindestens zwei Personen sitzen (es wäre natürlich auch eine Person vorstellbar).
- der dritte Term darf nur dann Null ergeben, wenn genau n (mit n = Anzahl der Profile) Einsen in der Matrix vorkommen. Dies stellt sicher, dass auch wirklich jede Person einen Sitzplatz hat.
- der vierte Term gibt die addierten Distanzen der an jedem Tisch versammelten Profile an.

Clusteranalyse

Eine andere Möglichkeit zur Lösung des Tischproblems bietet die Clusteranalyse. Mit ihrer Hilfe können Klassen von Objekten im mehrdimensionalen Merkmalsraum gefunden werden. Dabei sollen Objekte die zu einer Klasse gehören einander möglichst ähnlich sein (Homogenität), während Objekte unterschiedlicher Klassen einander möglichst unähnlich sein sollen (Heterogenität) (Nakhaeizadeh, 1998, S. 110). Beliebtes Anwendungsgebiet der Clusteranalyse ist u. a. die Marktsegmentierung. Hierbei werden die Kunden eines Unternehmens gemäß ihrer Vorlieben, ihres Alters, Geschlechts oder auch anderer Merkmale in Segmente (Cluster) eingeteilt. So können Werbeamaßnahmen an jedes Segment spezifisch angepasst werden.

Mit der Clusteranalyse ließen sich die Profile also mit relativ einfachen Mitteln in Gruppen einteilen. Nachfolgend soll zum besseren Verständnis der k-means-Algorithmus (Gerken, 2003, S. 18) erläutert werden, bei dem es sich um ein beliebtes Clustering-Verfahren handelt.

k-means

Um den k-means-Algorithmus anzuwenden, ist folgende Ausgangssituation nötig:

- es existieren n Objekte (im Beispiel die Profile)
- es wird eine Anzahl k der zu generierenden Cluster angegeben
- es existiert eine Abstandsfunktion (z. B. eine der Funktionen aus Abschnitt 3.3)
- es existiert eine Funktion zur Berechnung des Mittelpunktes eines Clusters

Anschließend verläuft der Algorithmus in folgenden Schritten:

1. es werden zufällig k Punkte als Clusterzentren bestimmt
2. jedes Objekt wird dem nächstgelegenen Zentrum zugeordnet
3. für jedes Cluster ist das Zentrum neu zu berechnen
4. Abbruch, falls sich die Zentren nicht mehr ändern; sonst wieder zu Schritt 2

Der k-means-Algorithmus

Fazit

Die Clusteranalyse mit dem k-means-Algorithmus ist mit geringem Aufwand umsetzbar und kann durch Definition einer individuellen Abstandsfunktion an verschiedene Anforderungen angepasst werden. Wohingegen sich der Ansatz, die Profile mit Hilfe von Hopfield-Netzen auf Tische zu verteilen, als kompliziert und aufwändig erweist. Somit verstößt er gegen das gut bewährte KISS-Prinzip[3]. Es wird also der Lösungsansatz mit Hilfe der Clusteranalyse bevorzugt.

[3] "Keep it simple, Stupid!"

4 Design und Implementierung

In diesem Abschnitt soll der Entwurf von get2gether dargestellt werden. Dabei wird zunächst von der groben Architektur des Systems ausgegangen, um in den weiteren Abschnitten in die Tiefe zu gehen und die Komponenten sowie das Klassenmodell vorzustellen.

4.1 Systemarchitektur

Die Anwendung soll als sog. Webanwendung realisiert werden. Die Applikationslogik wird somit auf einem Server ausgeführt der über ein Intranet oder über das Internet von einem Browser bedient werden kann. Dabei wird eine Zweischichtenarchitektur zugrunde gelegt, in der der Browser als Thin-Client fungiert und keinerlei Logik implementiert. Er dient lediglich als Schnittstelle zur Dateneingabe und zur Präsentation der Ergebnisse. Der Server hingegen ist verantwortlich für Entgegennahme und Ausführung der Aktionen, die der Client initiiert und ggf. für deren Beantwortung. Hierzu zählt jegliche Geschäftslogik, Datenhaltung sowie alle notwendigen algorithmischen Berechnungen. Es wäre auf der Serverseite eine weitere Schichtentrennung denkbar. Oftmals wird eine Trennung zwischen der Geschäftslogik und der Persistenzschicht realisiert. Da aber die Persistenz in dieser Version von get2gether keine Rolle spielt, wird hier bewußt auf eine weitere Unterteilung verzichtet.

Java

Aufgrund ihrer Plattformunabhägigkeit bietet sich die Sprache Java zur Implementierung von get2gether an. Dabei bleibt die Unabhängigkeit von der Hardware- und Betriebssystemplattform des Servers gewahrt. Es könnte sich folglich sowohl um einen Windows-Server als auch um einen beliebigen Unix-Server handeln. Des Weiteren erhöht diese Entscheidung die Flexibilität bei der Auswahl der Clients. So könnte z. B. eine Schnittstelle angeboten werden, mit deren Hilfe die Dienste des Servers durch ein Applet per RMI[1] aufgerufen werden könnten. Es wäre auch ein Handy als Client vorstellbar, auf dem die Clientsoftware mit Hilfe eines Midlets[2] realisiert ist.

[1] Remote Method Invocation

[2] Eine Anwendung für Mobiltelefone die dem MIDP-Standard entspricht

Um den eigenen Entwicklungsaufwand möglichst gering zu halten[3], sollen zudem einige Open-Source-Produkte im Rahmen dieser Arbeit genutzt werden. Da besagte Produkte - v. a. im Rahmen des Jakarta-Projektes von Apache[4] - Java als Entwicklungssprache nutzen, bietet es sich auch aus dieser Sicht heraus an.

Model-View-Controller

Wie in Kapitel 3 beschrieben, soll get2gether zunächst nicht konkret für PDAs umgesetzt werden. Um jedoch eine mögliche Portierung der Clients auf PDAs in Zukunft nicht unnötig zu erschweren, bietet sich ein Design nach dem Model-View-Controller Paradigma (MVC) an. Da MVC nicht das eigentliche Thema dieser Arbeit ist, soll auf das Konzept hier nicht im Detail eingegangen werden. Näheres kann sonst bei (Gamma u. a., 1996, S. 287ff) oder auch bei (Chan und Stegmann, 1997) nachgeschlagen werden. Zur einfacheren Umsetzung des MVC-Musters in Webanwendungen ist der Einsatz eines Frameworks sinnvoll, welches dem Entwickler viel Arbeit abnimmt. Zu diesem Zweck wurde das Struts-Framwork[5] entwickelt, das inzwischen als Quasi-Standard bei der Entwicklungen von Webapplikationen mit Java gilt. Abbildung 4.1 stellt den Arbeitsablauf einer mit Struts realisierten Webanwendung etwas vereinfacht dar.

1. Es wird ein HTML-Formular abgeschickt. Der Request wird an das *ActionServlet* geleitet, welches von Struts zur Verfügung gestellt wird.
2. Das *ActionServlet* “sieht“ in der Datei *struts-config.xml* nach, welche Aktion für den entsprechenden Request auszuführen ist.
3. Es wird die entsprechende Unterklasse der Struts-Klasse *Action* mit ihrer *execute*-Methode aufgerufen (in der Abbildung *DoSomethingAction*).
4. Die *Action*-Unterklasse ruft die eigentliche Anwendungslogik auf, welche z. B. die Formularwerte in einer Datenbank speichert.
5. Die *Action* teilt dem *Servlet* mit was es an den Browser zurückgeben soll (z. B. bei erfolgreichem Administrator-Login eine Admin-Seite und sonst eine Fehlerseite).
6. Das *ActionServlet* sendet die entsprechende Seite zum Browser.

[3] “Gute Informatiker sind faule Informatiker“ (Prof. Dr. Guido Pfeiffer)
[4] http://jakarta.apache.org
[5] http://struts.apache.org/

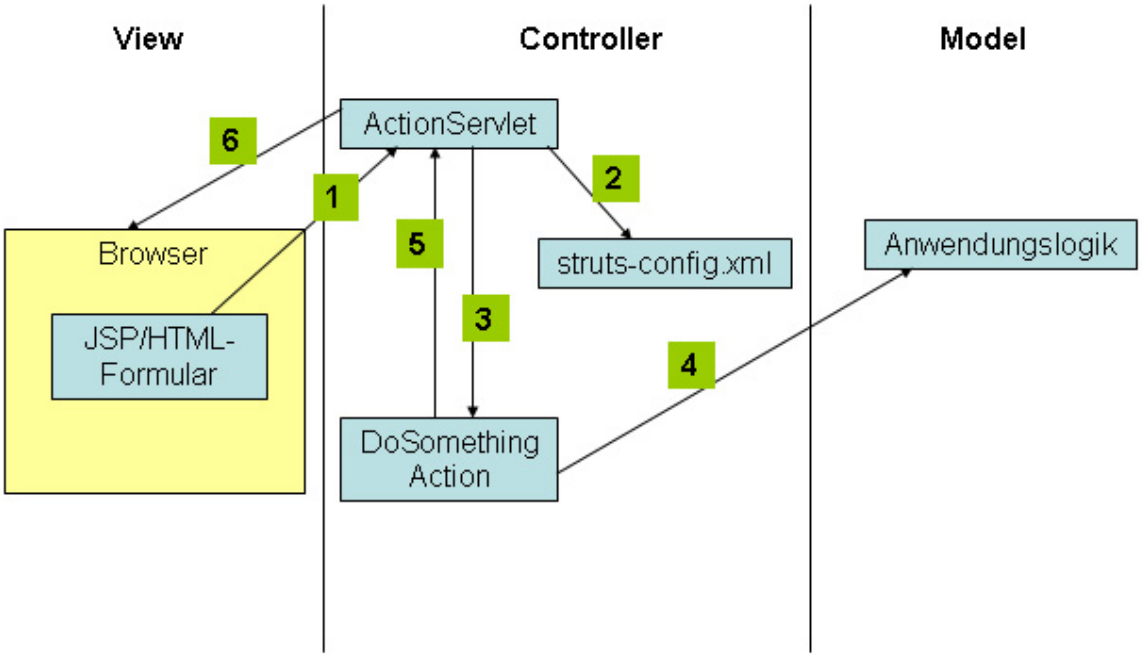


Abbildung 4.1: Vereinfachter Arbeitsablauf einer Webapplikation mit Struts

4.2 Komponenten

Die Anwendung besteht im Wesentlichen aus vier Komponenten. Zum Einen enthält sie JSP[6]-Seiten. Diese zählen im MVC-Konzept zum View-Teil. Sie werden zur Laufzeit in gewöhnliche HTML-Seiten umgewandelt, die an den Browser geschickt werden. Die JSPs dienen der Eingabe und Anzeige. Die nächste Komponente bildet die sog. *Web-Komponente*. Hier befinden sich die in 4.1 bereits geschilderten Controller-Klassen. Zu jedem in 3.2 beschriebenen Use-Case wurde eine entsprechende Action-Klasse entworfen. Sobald der Benutzer durch seine Eingaben einen bestimmten Use-Case auslöst, wird die entsprechende Action aktiviert. Auf der Seite der eigentlichen *Applikationslogik* ist nun noch zwischen zwei Komponenten zu unterscheiden. Zum Einen gibt es eine Komponente zur *Profilverwaltung*. Hier können Profilvorlagen angelegt werden. Anhand dieser Vorlagen können anschließend passende Profile eingegeben werden. Die andere Komponente der Applikationslogik bildet die *Clustering-Komponente*. Hier befinden sich die nötigen Klassen um die Profile zu clustern, also jedem Clubbesucher einen Tisch zuzuweisen.

[6] Java Server Pages

4.3 Klassenmodelle

Web-Komponente

Das Klassenmodell der Web-Komponente ist recht einfach. Da es sich aber um relativ viele (14) Klassen handelt, soll an dieser Stelle eine Beschreibung das UML-Modell ersetzen. Eine einfache Übersicht bietet dennoch Abb. 4.2. Die Klasse BaseAction erweitert die Klasse Action aus dem Struts-Framework und ist als Oberklasse aller Action-Klassen von get2gether gedacht. Sie bietet deshalb einige Konstanten und Methoden an, die von allen Unterklassen genutzt werden können. So hat sie z. B. eine Methode um zu überprüfen ob der aktuelle Benutzer als Administrator eingeloggt ist. Alle anderen Action-Klassen implementieren die Methode *execute()*. Diese wird vom Struts-Framework automatisch aufgerufen. Hier wird jeweils definiert was die Action konkret ausführen und was sie zurückgeben soll. Zu jedem Use-Case existiert mindestens eine Action-Klasse. Des Weiteren gibt es u. a. auch Action-Klassen zur Anzeige der Startseite oder zur Anzeige des Impressums.

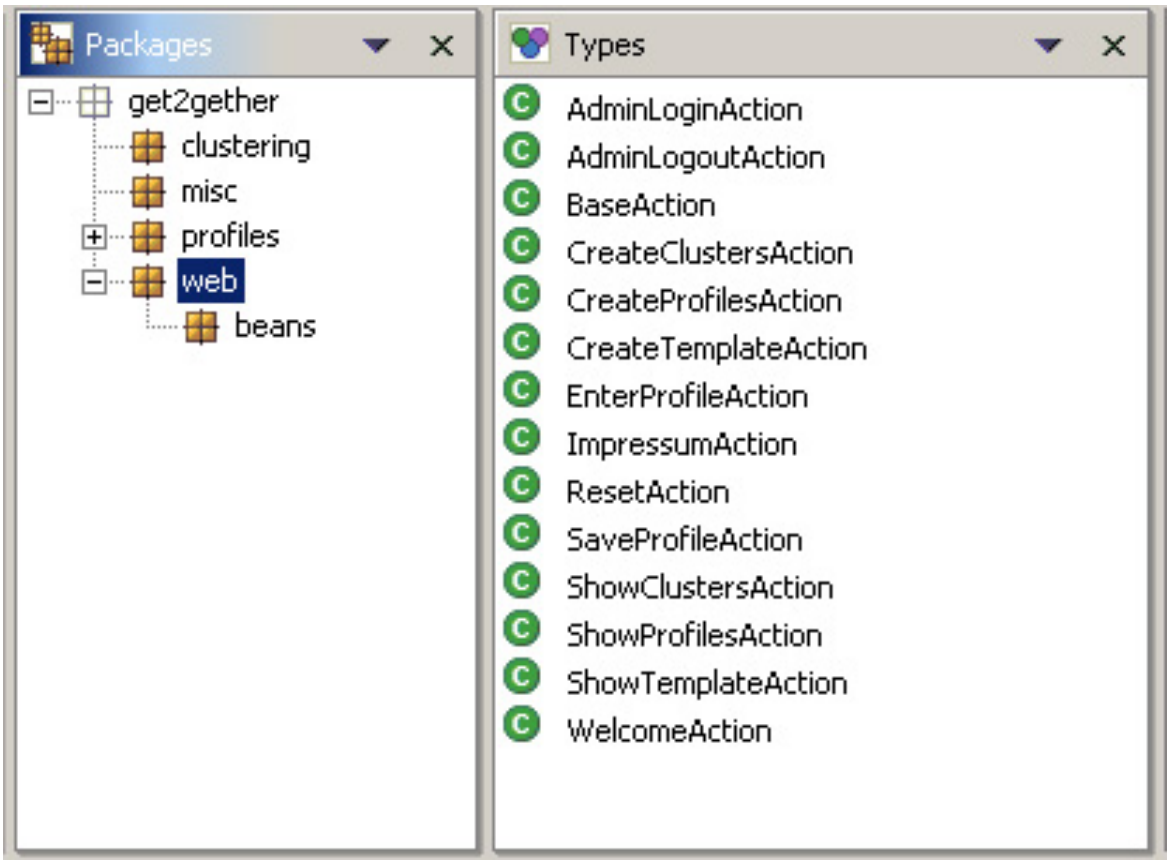


Abbildung 4.2: Die Action-Klassen der Web-Komponente

Komponente Profilverwaltung

Die Komponente Profilverwaltung, wie sie in Abb. 4.3 als Klassenmodell dargestellt ist, besteht im Wesentlichen aus den drei Klassen *ProfileManager*, *ProfileTemplate* und *Profile*. Mit Hilfe der Klasse ProfileManager lassen sich u. a. Profilvorlagen definieren, wie dies in 3.2 bereits geschildert wurde. Des Weiteren nimmt diese Klasse alle eingegebenen Profile entgegen und verwaltet diese in einer Liste. Es lassen sich auch Zufallsprofile erzeugen, was der Demonstration von get2gether dienen soll.

Eine Instanz der Klasse ProfileTemplate definiert den Rahmen für eine Serie von Profilen. Es kann also angegeben werden welche und wie viele Attribute ein Profil beinhalten soll. Für jedes Attribut wird zudem ein Wertebereich und ein Gewicht angegeben.

Nach Erzeugung eines ProfileManagers und eines ProfileTemplates können nun Instanzen der Klasse Profile erzeugt und dem ProfileManager übergeben werden. Jedes Profil hat einen Namen, der dem Namen des Feriengastes entsprechen sollte. Das Profil muss sich an seine Profilvorlage halten. D. h. für jedes in der Vorlage definierte Attribut muss das Profil einen ausgefüllten Wert enthalten der im zulässigen Wertebereich liegt.

Clustering-Komponente

Das Klassenmodell der Clustering-Komponente von get2gether ist in Abb. 4.4 dargestellt. Es wurden zunächst zwei Interfaces entwickelt. Das Interface *DistanceFunction* definiert die Methode *computeDistance()*. Diese soll die Distanz zwischen zwei übergebenen Profilen als Fließkommazahl zurückgeben. Dabei sind verschiedene Berechnungsverfahren vorstellbar. Einige Möglichkeiten wurden bereits in Abschnitt 3.3 erörtert. Die öffentliche Schnittstelle für die eigentliche Clustering-Klasse wird vom Interface *Clusterer* definiert. Dieses Interface erwartet die Übergabe einer DistanceFunction, mit deren Hilfe die jeweiligen Distanzen berechnet werden. Des Weiteren wird mit der Methode *setNoOfClusters()* die Anzahl der gewünschten Cluster (bzw. konkret bei get2gether die Anzahl der Tische) angegeben. Schließlich wird der Methode *clusterProfiles()* eine Liste der Profile übergeben, die in Cluster eingeteilt werden sollen. Zurückgegeben wird eine Art Dictionary[7] das als Schlüssel die Nummer des Clusters und als Wert eine Liste mit den diesem Cluster zugeordneten Profilen enthält.

Durch die Definition der beiden Interfaces sind verschiedenste Implementierungen von Distanzfunktionen und von Clusterern denkbar. Es wird ein flexibles Modell erreicht, in dem die Algorithmen leicht ausgetauscht und an neue Bedürfnisse angepasst werden können.

[7] im Javajargon ein Objekt vom Typ *Map*

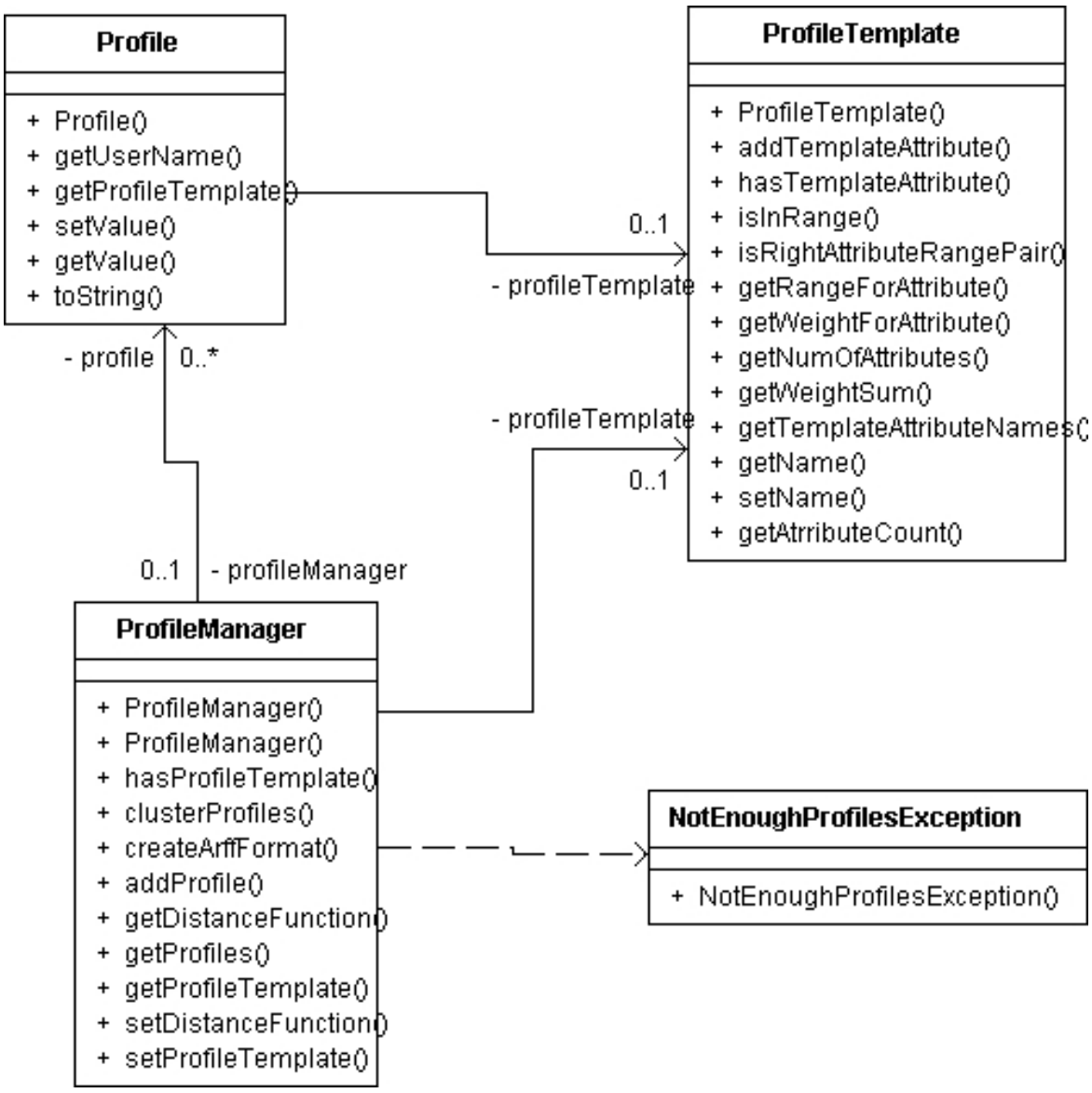


Abbildung 4.3: Klassenmodell der Komponente Profilverwaltung

Konkret wurde für jedes Interface je eine Implementierung realisiert. Die Klasse *WeightedEuklidDistanceFunction* berechnet die Distanz als gewichtete Euklid-Distanz, wie dies bereits in Abschnitt 3.3 beschrieben wurde.

Die Klasse *KMeansClusterer* implementiert den k-means-Algorithmus, wie er in Abschnitt 3.4 erläutert wurde. Dabei benutzt der Algorithmus die gewichtete Euklid-Distanzfunktion um den jeweiligen Abstand zwischen zwei Profilen zu berechnen.

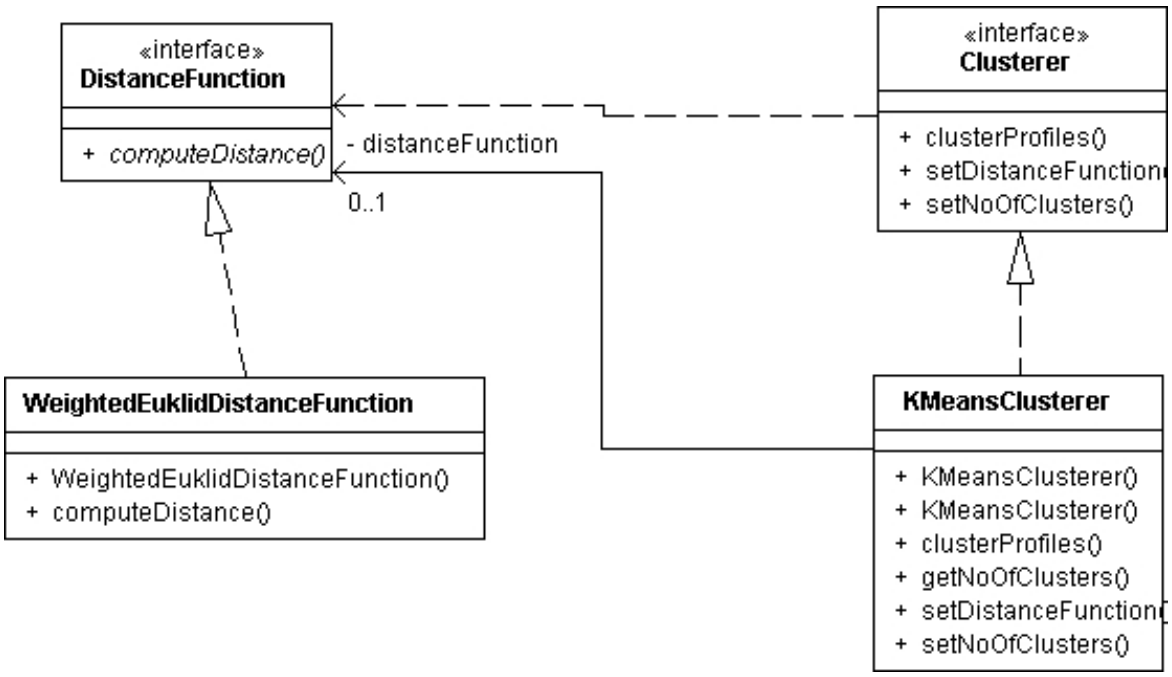


Abbildung 4.4: Klassenmodell der Clustering-Komponente

weka

Weka[8] ist eine Sammlung von Algorithmen für Data-Mining-Zwecke. Das Framework wird an der Universität von Waikato, Neuseeland entwickelt, ist in Java implementiert und als Open-Source frei erhältlich. In einer früheren Version von get2gether wurde der k-means-Algorithmus aus dem weka-Paket genutzt. Leider ließ die weka-Implementierung aber keine Gewichtung der Attribute zu, wie es von get2gether gefordert wurde. Auch eine Anpassung oder Erweiterung der Implementierung erwies sich als zu aufwändig. So wurde der k-means-Algorithmus vom Autor implementiert.

[8] http://www.cs.waikato.ac.nz/ ml/weka/

5 Fallstudie

In diesem Kapitel soll die Anwendung get2gether anhand einer Fallstudie mit Hilfe von Screenshots näher vorgestellt werden.

5.1 Startseite

Zunächst befinden wir uns auf der Startseite (s. Abb. 5.1). In dieser Phase könnte ein möglicher Benutzer sein Profil noch nicht eingeben, da bislang keine Profilvorlage erstellt wurde. Um dies zu tun, loggen wir uns als Admin ein.

5.2 Administrationsmenü

Es erscheint die Administrationsseite, wie in Abb. 5.2 zu sehen. An dieser Stelle hat man die Möglichkeit eine Profilvorlage zu erstellen und sie sich anzeigen zu lassen, Zufallsprofile zu Demonstrationszwecken erzeugen und sich alle vorhandenen Profile anzeigen zu lassen oder die Profile in Cluster einteilen zu lassen. Wir wollen zunächst eine Profilvorlage mit fünf Attributen anlegen.

5.3 Profilvorlage

Es erscheint eine Seite (s. Abb. 5.3)auf der fünf Attribute mit Wertebereich und Gewichtung angegeben werden können. Wir füllen also fünf Attribute aus. Der Wertebereich liegt jeweils zwischen Eins und Zehn. Da uns besonders am Herzen liegt, was die Clubgäste von Tennis halten, gewichten wir dieses Attribut besonders schwer und zwar mit 60%. Alle anderen Attribute erhalten eine Gewichtung von 10%.

Abbildung 5.1: Die Startseite von get2gether

5.4 Profil ausfüllen

Nachdem die Profilvorlage erstellt ist, können wir nun Profile eingeben. Dazu loggen wir uns als Admin aus und klicken auf der Startseite auf den Link “Profil eingeben“. Es erscheint die Eingabeseite wie in Abb. 5.4. Wir füllen also ein Profil aus und speichern es.

Abbildung 5.2: Die Administrationsseite von get2gether

5.5 Zufallsprofile erzeugen und alle Profile anzeigen

Nun loggen wir uns wieder als Admin ein. Wir lassen uns noch 20 weitere Profile automatisch erzeugen. Diese Funktionalität ist nicht für den praktischen Einsatz gedacht und wurde deshalb nicht in Abschnitt 3.2 erwähnt. Sie dient vielmehr der Demonstration der Anwendung. So müssen nicht diverse Profile von Hand eingegeben werden. Der Admin gibt also die Anzahl der gewünschten Profile an, und prompt werden ebenso viele Profile mit einem

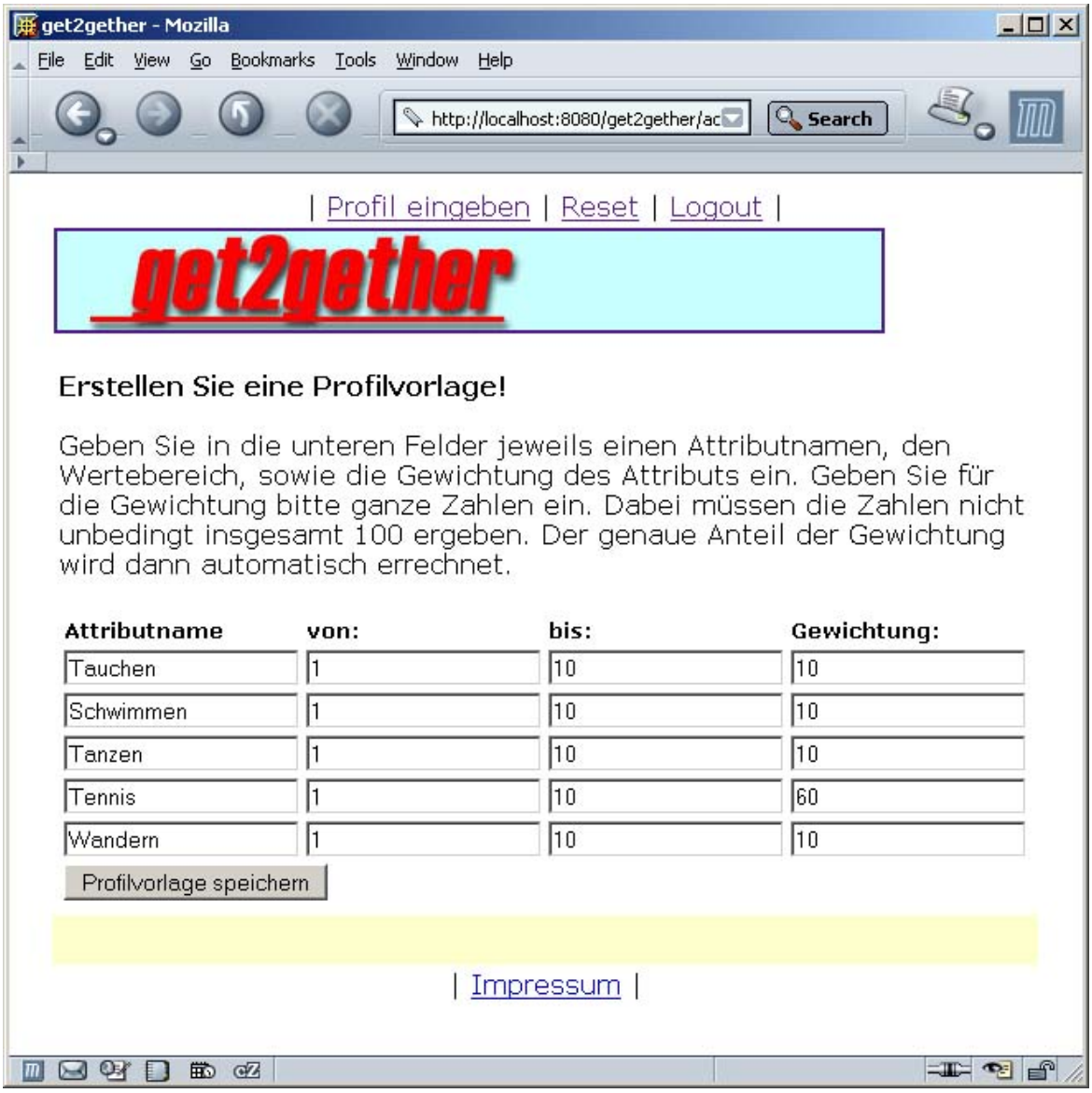


Abbildung 5.3: Eine ausgefüllte Profilvorlage

Zufallsnamen und einem zu jedem Attribut passenden Zufallswert erzeugt. Diese Profile sind natürlich gültig gemäß der vorher erstellten Profilvorlage.

Anschließend lassen wir uns alle Profile anzeigen, wie in Abb. 5.5 zu sehen. Unser eingegebenes Profil ist an erster Stelle zu sehen. Es folgen die 20 generierten Profile mit zufälligen Namen und zufälligen Attributwerten, die jeweils im zulässigen Wertebereich liegen.

Abbildung 5.4: Ein ausgefülltes Profil

5.6 Profile in Cluster einteilen

Zu guter Letzt wollen wir uns alle Profile in Cluster einteilen lassen. Wir geben an, dass die Profile auf vier Cluster zu verteilen sind. Anschließend sehen wir - wie in Abb. 5.6 - eine Übersicht aller Cluster und der ihnen zugeordneten Profile. Dabei ist schon beim flüchtigen Blick erkennbar, dass innerhalb jedes Clusters die Werte für das Attribut Tennis deutlich enger zusammenliegen als bei den anderen Attributen.

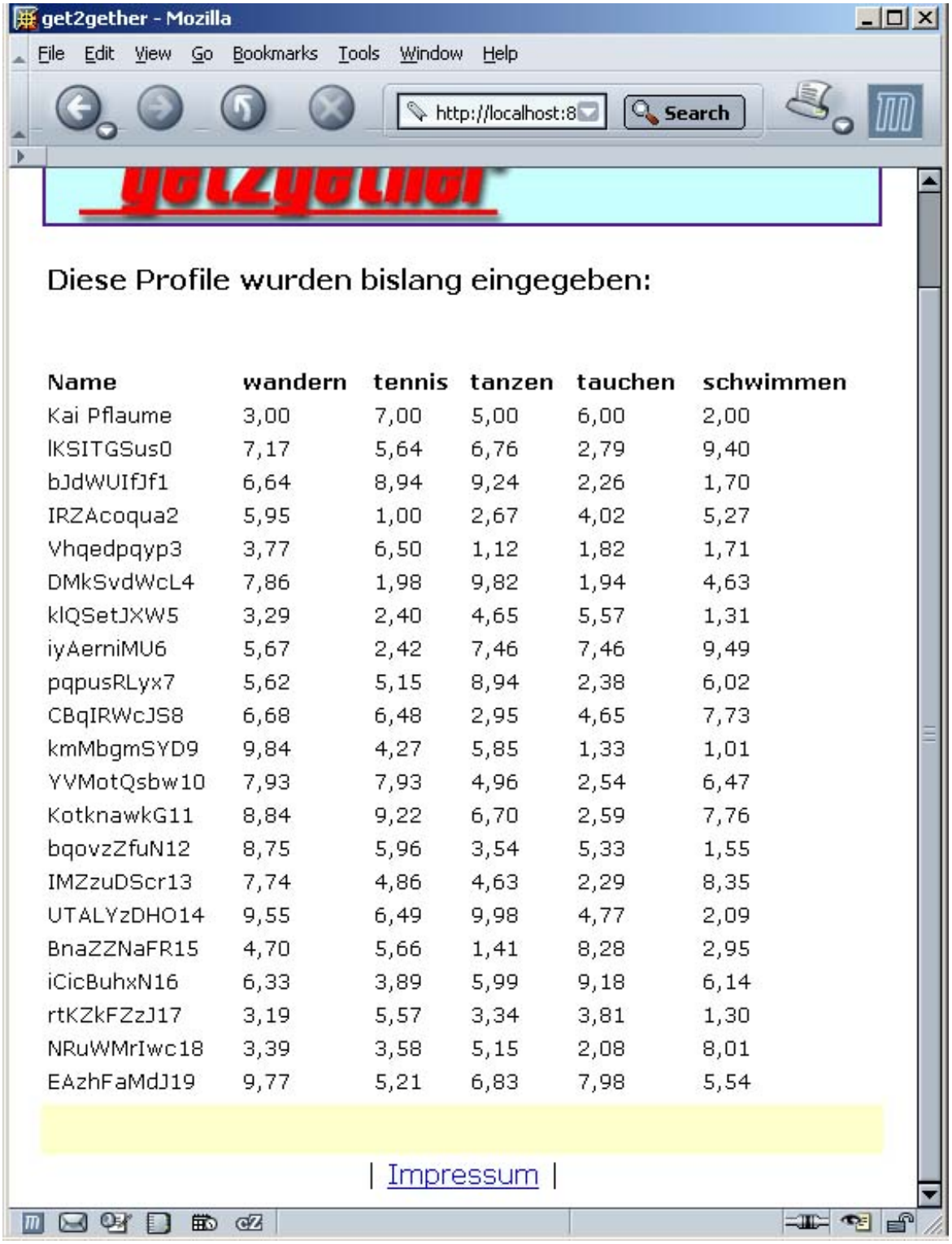
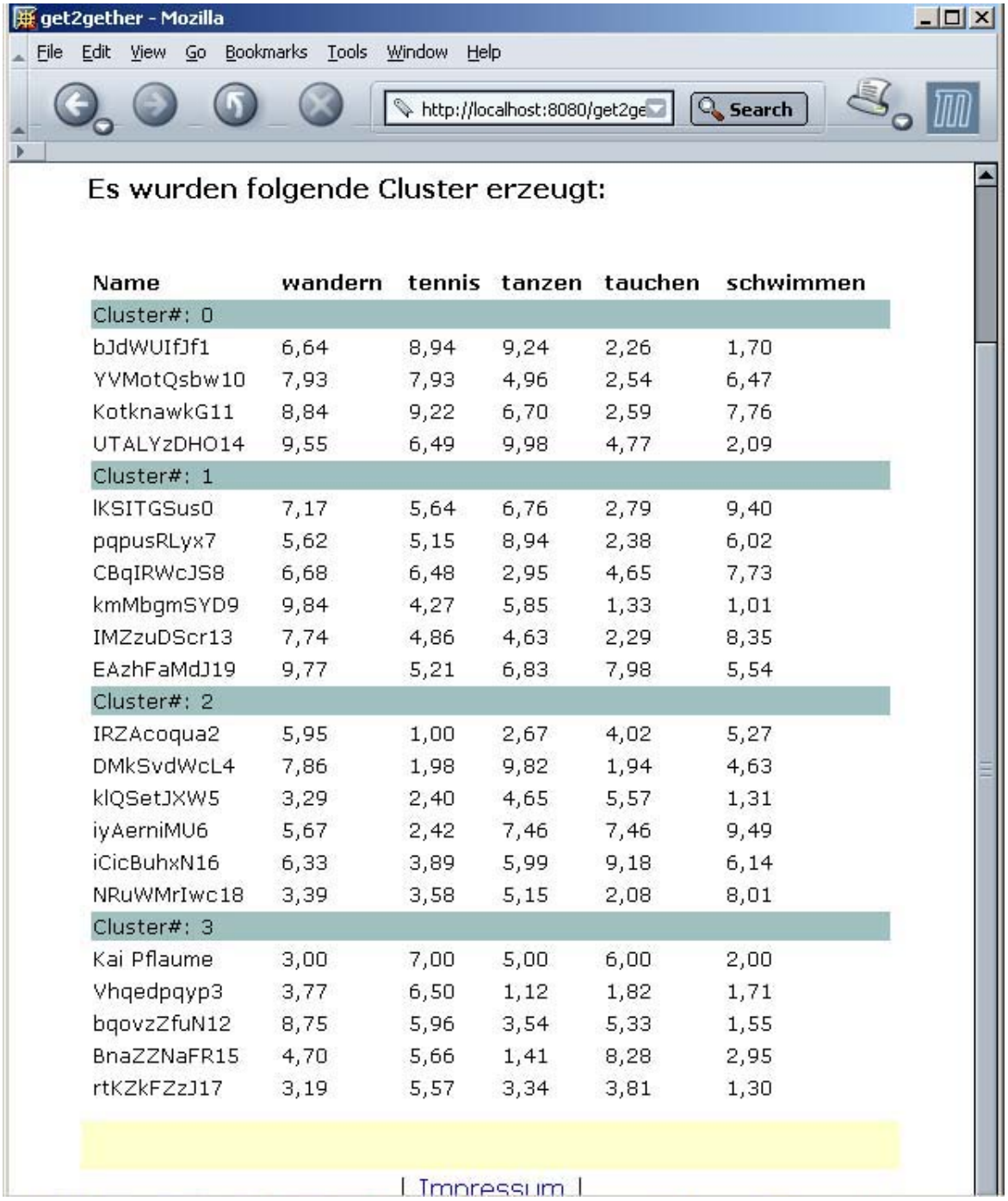


Abbildung 5.5: Übersicht aller eingegebenen Profile



Es wurden folgende Cluster erzeugt:

Name	wandern	tennis	tanzen	tauchen	schwimmen
Cluster#: 0					
bJdWUIfJf1	6,64	8,94	9,24	2,26	1,70
YVMotQsbw10	7,93	7,93	4,96	2,54	6,47
KotknawkG11	8,84	9,22	6,70	2,59	7,76
UTALYzDHO14	9,55	6,49	9,98	4,77	2,09
Cluster#: 1					
lKSITGSus0	7,17	5,64	6,76	2,79	9,40
pqpusRLyx7	5,62	5,15	8,94	2,38	6,02
CBqIRWcJS8	6,68	6,48	2,95	4,65	7,73
kmMbgmSYD9	9,84	4,27	5,85	1,33	1,01
IMZzuDScr13	7,74	4,86	4,63	2,29	8,35
EAzhFaMdJ19	9,77	5,21	6,83	7,98	5,54
Cluster#: 2					
IRZAcoqua2	5,95	1,00	2,67	4,02	5,27
DMkSvdWcL4	7,86	1,98	9,82	1,94	4,63
klQSetJXW5	3,29	2,40	4,65	5,57	1,31
iyAerniMU6	5,67	2,42	7,46	7,46	9,49
iCicBuhxN16	6,33	3,89	5,99	9,18	6,14
NRuWMrIwc18	3,39	3,58	5,15	2,08	8,01
Cluster#: 3					
Kai Pflaume	3,00	7,00	5,00	6,00	2,00
Vhqedpqyp3	3,77	6,50	1,12	1,82	1,71
bqovzZfuN12	8,75	5,96	3,54	5,33	1,55
BnaZZNaFR15	4,70	5,66	1,41	8,28	2,95
rtKZkFZzJ17	3,19	5,57	3,34	3,81	1,30

| Impressum |

Abbildung 5.6: Übersicht aller geclusterten Profile

6 Ausblick

In diesem Abschnitt soll auf einige Punkte eingegangen werden, die in dieser Version von get2gether nicht umgesetzt wurden, die jedoch in eine “produktreife“ Version Eingang finden müßten.

6.1 PDA

Für den endgültigen Einsatz in einem Ferienclub sollte die Anwendung auf PDAs an die Feriengäste verteilt werden, wie dies im Szenario in Abschnitt 2.1 vorgestellt wurde. Aufgrund der Architektur von get2gether, das als Client-Server-Anwendung unter Einsatz des MVC-Paradigmas realisiert ist, ließe sich relativ einfach eine alternative Oberfläche umsetzen.

6.2 Sicherheit

Der Aspekt der Sicherheit wurde in dem entwickelten Prototypen wenig beachtet. Benutzer müssen sich bislang nicht anmelden und es kann im Prinzip jeder der am Terminal vorbeikommt ein Profil eingeben. In einer späteren Version müsste natürlich eine Benutzerverwaltung eingesetzt werden. Die Benutzer müßten sich am Terminal zunächst authentifizieren um Zugang zur Profileingabe zu erhalten.

Des weiteren sollte der Datenverkehr zwischen Client und Server u. U. verschlüsselt erfolgen, damit er nicht abgehört werden kann. Hierzu bietet sich das Protokoll HTTPS[1] an, bei dem die Verbindung zwischen Client und Server verschlüsselt wird.

[1] Hypertext Transfer Protocol Secure

6.3 Persistenz

Die eingegebenen Profile der Clubgäste werden bislang nicht gespeichert. Wird der Server also neu gestartet, so sind alle Informationen verloren. In einer späteren Version sollte also ein Persistenzmechanismus zum Einsatz kommen. Die Profile könnten dann in einer Datenbank oder auch in XML-Dateien gespeichert werden.

Literaturverzeichnis

[Babic 2003] BABIC, Alexander: *Entwicklung einer profilverarbeitenden ubiquitären Anwendung*, Hochschule für angewandte Wissenschaften Hamburg, Diplomarbeit, 2003

[Callan 2003] CALLAN, Robert: *Neuronale Netze im Klartext*. 1. Auflage. Pearson Studium, 2003

[Chan und Stegmann 1997] CHAN, Tin-Ho ; STEGMANN, Gerhard: *Ausarbeitung zum MVC-Framework*. 1997. – URL `http://www.informatik.fh-muenchen.de/~schieder/seminar-oo-modellieren-ws97-98/f7alpha1.informatik.fh-muenchen.de/~ifw94032/index.html`. – Zugriffsdatum: 2004-11-21

[Citypopulation] CITYPOPULATION: *Deutschland - Städte & Provinzen - Einwohnerzahlen & Karten*. – URL `http://www.citypopulation.de/Deutschland_d.html`. – Zugriffsdatum: 2004-10-01

[Gamma u. a. 1996] GAMMA, Erich ; HELM, Richard ; JOHNSON, Ralph ; VLISSIDES, John M.: *Entwurfsmuster - Elemente wiederverwendbarer objektorientierter Software*. 5. Auflage. Addison-Wesley, 1996

[Gerken 2003] GERKEN, Wolfgang: *Analytische Informationssysteme, Vorlesungsskript, WS03/04, Kap. 6 Álgorithmen beim Data Mining´*. 2003

[Görz u. a. 2003] GÖRZ, Günther ; ROLLINGER, Claus-Dieter ; SCHNEEBERGER, Josef: *Handbuch der Künstliche Intelligenz*. 4. Auflage. Oldenbourg Wissenschaftverlag, 2003

[Lämmel und Cleve 2001] LÄMMEL, Uwe ; CLEVE, Jürgen: *Lehr- und Übungsbuch Künstliche Intelligenz*. 1. Auflage. Fachbuchverlag Leipzig, 2001

[MAV-Online] MAV-ONLINE: *MAV-Online*. – URL `http://www.mav-online.de/O/121/Y/67218/VI/10064272/default.aspx`. – Zugriffsdatum: 2005-01-21

[Nakhaeizadeh 1998] NAKHAEIZADEH, Gholamreza: *Data Mining Theoretische Aspekte und Anwendungen*. 1. Auflage. Physica Verlag Heidelberg, 1998

[Net-Lexikon] NET-LEXIKON: *Euklidischer Abstand*. – URL `http://www.lexikon-definition.de/Euklidische-Distanz.html`. – Zugriffsdatum: 2004-09-26

[Werner 2004] WERNER, Bodo: *Mathematik für das Lehramt an der Grund- und Mittelstufe sowie an Sonderschulen Teil II (Lineare Algebra und analytische Geometrie): SoSe 04*. UNI Hamburg, 2004. – URL `http://www.math.uni-hamburg.de/home/werner/GruMiSoSe04TeilII.pdf`. – Zugriffsdatum: 2004-09-28

[Worldatlas] WORLDATLAS: *Worldatlas - Top 100 Cities of the World*. – URL `http://www.worldatlas.com/citypops.htm`. – Zugriffsdatum: 2004-10-01

[Zadeh 1965] ZADEH, Lotfi A.: Fuzzy Sets. In: *Journal of Information and Control* 8 (1965), S. 338–353

[Zell 2003] ZELL, Andreas: *Simulation neuronaler Netze*. 4. Auflage. Oldenbourg Wissenschaftverlag, 2003