ETH Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Student Thesis SA-2005-08
Winter Term 2004/2005

Authors:
Schifferle Sandro D-ITET
Braun Christian D-ITET

Tutors:
Matthias Bossardt
Vincent Lenders
Dr. Martin May

Supervisor:
Prof. Dr. Bernhard Plattner

9.2.2005

BlueDating

Dating Application for Bluetooth Enabled Mobile Phones

TIK Institut für Technische Informatik und Kommunikationsnetze
Computer Engineering and Networks Laboratory

Abstract

Deutsch

Dieses Dokument beschreibt unsere Semesterarbeit am *Institut für Technische Informatik und Kommunikationsnetze (TIK)* [16] des *Departements Informationstechnologie und Elektrotechnik (D-ITET)* [17] der *ETH Zürich* [18].
Die Aufgabe unserer Semesterarbeit war es, eine Dating Applikation namens BlueDating für Java unterstützende Mobiltelefone, welche über Bluetooth kommuniziert, zu entwickeln. Andere vergleichbare Applikationen, welche sich derzeit in der Entwicklung oder auf dem Markt befinden, sind gerätespezifisch programmiert (in der Regel für das Symbian Betriebssystem von Nokia). Hier kann BlueDating, dank Java-Technologie, eine breitere Unterstützung über Herstellergrenzen hinweg bieten.
Das Konzept von BlueDating wurde so erstellt, dass eine Kommerzialisierung in verschiedenen Varianten möglich ist. Ausserdem wurde Wert auf die Berücksichtigung der Privatsphäre der Benutzer gelegt, so dass persönliche Daten von aussen nicht abrufbar sind.
Unsere Applikation wurde erfolgreich auf dem *Nokia 6630* getestet.

English

This document describes our semester thesis at the *Computer Engineering and Networks Laboratory (TIK)* [16] of the *Department of Information Technology and Electrical Engineering (D-ITET)* [17] at *ETH Zurich* [18].
The goal of our semester thesis was to write a dating application named BlueDating for java enabled mobile phones that communicates over Bluetooth radio. Other comparable applications in development or on the market are programmed specifically for certain devices (usually for the Symbian operating system by Nokia). In this respect BlueDating can support a wider range of devices by various manufacturers, thanks to Java technology.
The concept of BlueDating was designed to facilitate the introduction of commercial elements in various ways. Also, privacy issues were kept in mind by preventing the disclosure of personal data at all times.
Our application has been successfully tested on the *Nokia 6630*.

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

Kapitel 1

Einleitung

In der heutigen pluralistischen Gesellschaft sehen sich zahlreiche Jugendliche mit Problemen konfrontiert, unbekannte Menschen kennenzulernen. In der Unterschiedlichkeit der Ansichten und Geschmäcker, in der kleinerwerdenden Überlappung der Normensysteme der einzelnen Individuen, fällt es oft immer schwerer, eine gemeinsame Basis für den Beginn einer Konversation zu finden. Unverständnis und Enttäuschungen lassen die Scheu und Unsicherheit zur nahezu unüberwindbaren Hürde anwachsen. Die verstohlenen Blicke in der Disco, die - im Rätseln über die passende Ansprachemöglichkeit - dann doch keine weiteren Folgen oder gar Erfolge mit sich bringen, mehren sich.
Dating-Annoncen und -Webseiten erleben in diesem Umfeld grossen Andrang, doch es besteht Potential, diese Art von Diensten zu verbessern. Mit BlueDating unternimmt die rechnergestützte elektronische Partnersuche den Schritt auf die Strasse und hilft im täglichen Leben, die zwischenmenschlichen Hürden zu überwinden: Einer Dating-Applikation auf dem Mobiltelefon (einem Gerät das heute sowieso beinahe jeder und jede auf sich trägt) welche ständig dabei und auf Empfang ist, der dürfte der Traumpartner oder die Traumpartnerin nicht entgehen! Dieser Vision bringt uns BlueDating einen guten Schritt näher.
Um auch den richtigen Traumprinzen oder die richtige Traumprinzessin zu finden wird das BlueDating-Programm mit detaillierten Informationen über Besitzer(in) sowie über die Eigenschaften des idealen Partners bzw. der idealen Partnerin gefüttert. Sind diese Angaben sorgfältig gewählt, findet BlueDating jemanden, der passt und die Chancen stehen gut, dass nicht nur für das erste Gespräch eine solide gemeinsame Basis besteht, sondern die Beziehung mit der gefundenen Person eine gute Chance hat.
So kompliziert es zwischen Männern und Frauen manchmal auch sein mag, so ist es auch sprachlich - daher werden wir uns im weiteren Verlauf dieser Dokumentation auf Begriffe wie "der Benutzer" oder "der Partner" beschränken, womit wir natürlich jeweils die männliche wie die weibliche Form meinen.
Im folgenden Kapitel werden aktuelle Projekte vorgestellt, die mit BlueDating vergleichbar sind. Kapitel 3 beschreibt die Java 2 Micro Edition. In Kapitel 4 wird Bluetooth vorgestellt, wobei auch die Java Implementierung von Bluetooth angesprochen wird. Das Kapitel 5 zeigt das Konzept von BlueDating auf. Unter Konzept wird das verstanden, was der Benutzer von der Software mitbekommt. Kapitel 6 wird die Implementation der Software detailliert erklären. In Kapitel 7 werden schlussendlich die erreichten Resultate aufgezeigt, Probleme angetönt und Aussichten auf Erweiterungen angesprochen.

Im Anhang A befindet sich eine Übersicht der eingebauten Debug Möglichkeiten, die man für Test- und Demonstrationszwecke nutzen kann. Im Anhang B wird näher auf die genutzte Entwicklungsumgebung eingegangen. Der Anhang C beschreibt den Inhalt der mitgelieferten CD-ROM. Im Anhang D findet sich die offizielle Aufgabenstellung. Anhang E zeigt den Zeitplan der Semesterarbeit und zu guter Letzt befindet sich im Anhang F ein Verzeichnis aller verwendeten Abkürzungen.

Kapitel 2

Verwandte Projekte

2.1 Internetbasierte Partnersuche

Diverse Webseiten versuchen bei der Partnersuche behilflich zu sein (z.B. [19] [20] [21]). Üblicherweise können Inserate geschrieben werden, welche sich meist beim Lesen filtern lassen, so dass beispielsweise nur Inserate einer gesuchten Altersklasse angezeigt werden. Teilweise ist auch eine Anmeldung, verbunden mit der Eingabe eines umfassenden Profils, nötig. Eine umfassende Studie über die internetbasierte Partnersuche wurde von Dr. des. Evelina Bühler-Ilieva am Soziologischen Institut der Universität Zürich durchgeführt [15].

2.2 Ortung und Detektion von Interaktionen

Generell ordnet sich das Projekt BlueDating ebenfalls inmitten von Technologien ein, die verwendet werden, um Geräte bzw. deren Besitzer zu orten und diese Information weiter zu verarbeiten. So gibt es schon seit über 15 Jahren Systeme, welche Mitarbeiter mittels ihres Badges in einem Gebäude lokalisieren, um dann automatisch Telefonanrufe zu ihnen weiterzuleiten (*ActiveBadge*, *ParcTab*, *Bat*). Auch die Ortung von Mobiltelefonen mittels der Auswertung von Daten der GSM-Antennen ist bereits weit verbreitet, diese wird dann beispielsweise verwendet, um den Benutzer mit Informationen über seine Umgebung (z.B. Restaurants, Geldautomaten oder Tankstellen in der Nähe) zu versorgen. Eine umfangreiche Liste von Projekten aus diesem Gebiet findet sich in [14]. Nachfolgend sind die wichtigsten verwandten Projekte vorgestellt.

2.3 Lovegety

Das erste weitläufig bekannt gewordene Gerät, welches über Funk nach potentiellen Dating-Partnern sucht, ist das "Lovegety". Dieses Gerät stammt von der Firma *Erfolg*, welche zuvor die allseits bekannten virtuellen Haustiere namens "Tamagotchi" auf den Markt gebracht hatte. 1998 kam Lovegety in Japan für 2900 Yen (umgerechnet gut 30 Schweizer Franken) auf den Markt. In Japan unter Teenagern ein Verkaufsschlager, fanden die Geräte im Rest der Welt jedoch kaum Verbreitung.
Die ovalen Geräte wurden in zwei Farben verkauft, rosa für Mädchen und hellblau für Jungen. Sie kommunizieren auf 300 MHz, wobei die Reichweite rund fünf Meter beträgt. Über drei einstellbare Modi kann der Besitzer seine Vorstellungen über

Abbildung 2.1: *Der Klassiker: Lovegety*

die Gestaltung des ersten Dates bekanntgeben: "talk", "karaoke" oder "get2", wobei die Auffassungen darüber auseinandergehen, was letzteres alles umfassen kann. Zwei Anzeigen informieren einerseits darüber, ob ein Lovegety von anderer Farbe in Reichweite ist, andererseits ob die eingestellten Modi übereinstimmen.

2.4 Serendipity

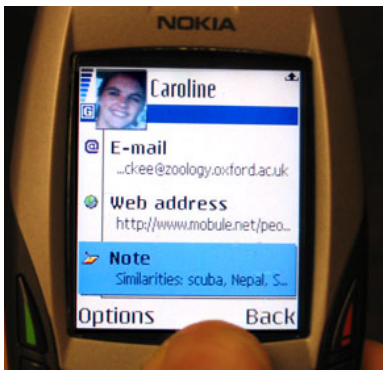


Abbildung 2.2: *Serendipity*

"Serendipity" [14] wird von Nathan Eagle, einem Doktorand an der *Human Dynamics Group* des *MIT Media Lab*, in einer Zusammenarbeit mit *Nokia* entwickelt. Auf der offiziellen Seite [22] sind kaum Informationen über das System vorhanden, etwas mehr verraten die unter "Press" aufgeführten Webseiten. Die Software befindet sich noch in Entwicklung.
Es handelt sich bei Serendipity um eine Applikation für die Symbian Serie 60 von Nokia, welche über Bluetooth kommuniziert. Durch regelmässige Suche über Bluetooth stellt das Programm fest, wenn sich ein anderes Gerät mit installierter Seredipity-Software in der Nähe befindet. Daraufhin überträgt es eine Identifikationsnummer des anderen Geräts an einen Server, auf dem dann eingegebene Profile der beiden Benutzer verglichen werden. Stimmen die Interessen genügend überein, so veranlasst der Server eine Benachrichtigung der beiden Benutzer. Das Programm kennt verschiedene Modi, so dass es neben der Freizeit auch bei der Kontaktknüpfung während der Arbeit hilfreich sein soll.

2.5 Phunkz

Abbildung 2.3: *Phunkz*

Auch dies eine Software, welche über Bluetooth Verbindungen aufnimmt und bei der Partnersuche helfen soll. "Phunkz" [23] ist jedoch wie Serendipity auf gewisse Mobilgeräte mit Symbian OS beschränkt, nämlich die Geräte der Nokia Serie 60 (und dazu kompatible) sowie UIQ (Symbian-Geräte mit Touchscreen und Stiftbedienung). Laut dem Forum auf der Phunkz-Webseite ist eine Portierung nach Java geplant. Die Software befindet sich noch in Entwicklung, eine Vorabversion stand im Dezember 2004 kurzzeitig zum Download im Internet.
Um festzustellen, wie gut potentielle Partner zueinander passen, vergleicht Phunkz einerseits jeweils ein Profil mit grundlegenden Angaben mit der Suchanfrage des anderen Geräts sowie Übereinstimmungen in der Beantwortung eines Katalogs von Ja/Nein-Fragen, wobei bei letzterem keine 100%ige Übereinstimmung erforderlich ist.

2.6 Spotme

Abbildung 2.4: *Spotme*

"Spotme" [24] ist ein Linuxbasiertes Gerät mit grossflächigem LC-Bildschirm, welches an Konferenzteilnehmer ausgehändigt wird. Diese können sich damit nicht nur über die anderen Anwesenden (im näheren Umkreis oder generell) informieren, das System kann auch benutzt werden um Nachrichten auszutauschen oder jemanden um ein Treffen zu ersuchen und gegenseitig virtuelle Visitenkarten auszutauschen. Ebenso kann das Gerät zur Zutrittskontrolle verwendet werden und allgemeine Informationen (z.B. das Tagesprogramm) lassen sich abrufen. Dieses System ist auf dem Markt erhältlich und kann auch für Konferenzen gemietet werden.
Zum System gehören neben den mobilen Geräten eine Serverinfrastruktur und stationäre Antennen, welche nicht nur den Mobilgeräten den Zugriff auf den Server ermöglichen, sondern auch Meldungen zwischen den Mobilgeräten weiterleiten

Abbildung 2.5: *Das mobile Gerät des Spotme-Systems*

können. Neben der Funkverbindung kann zwischen den Geräten, beispielsweise zum Übertragen einer Visitenkarte, eine Infrarotverbindung aufgebaut werden.

2.7 nTAG

Abbildung 2.6: *nTag*

Ähnlich wie Spotme funktioniert das "nTAG". Dieses Badge für Konferenzen entstand wie Serendipity im MIT Media Lab. Die Geräte kommunizieren über RFID mit einem Server und über Infrarot untereinander. Die Firma *nTAG Interactive* [25] vermietet die entsprechenden Systeme.

2.8 Weitere Projekte

Neben Serendipity und Phunkz sind weitere verwandte Applikationen in den Startlöchern:

- "BEDD" [26], welches auf Symbian-Geräten läuft und neben Dating auch Flohmarkt- und Chatfunktionalität bieten soll
- "SmartDater" [27] für Nokia Serie 60

- ebenso "Proxidating" [28]
- "CrowdSurfer" [29] von SmartPlanet in Verbindung mit einer Web-Plattform
- ebenso "smile dating" [30]
- "mobiluck" [31] für Symbian-Geräte und PDAs
- "BuZZone" [32] für PDAs und PCs
- "Speck" [33], das ein Standalone-Gerät werden soll

Abbildung 2.7: *BEDD auf einem Nokia nGage*

Abbildung 2.8: *Das Gerät von Speck*

Kapitel 3

Java 2 Micro Edition (J2ME)

3.1 Die Java Plattformen

Java bietet eine gemeinsame Plattform für die verschiedensten Architekturen, so dass Software unabhängig von der darunterliegeneden Hardware lauffähig ist. Die Software muss als einzige Bedingung diese Plattform unterstützen. Die Weiterentwicklung von Java vergrösserte die mitgelieferten *Packages* derart, dass es besonders wichtig wurde, eine Plattform für eingebette Systeme zu entwickeln, welche auf deren Einschränkungen abgestimmt ist. Diese neue Plattform namens *Java 2 Micro Edition* ist eine abgespeckte Version von der *Java 2 Standard Edition*, welche für Desktop Rechner enwickelt wurde. Für Server Anwendungen wurde speziell die *Java 2 Enterprise Edition* entwickelt.

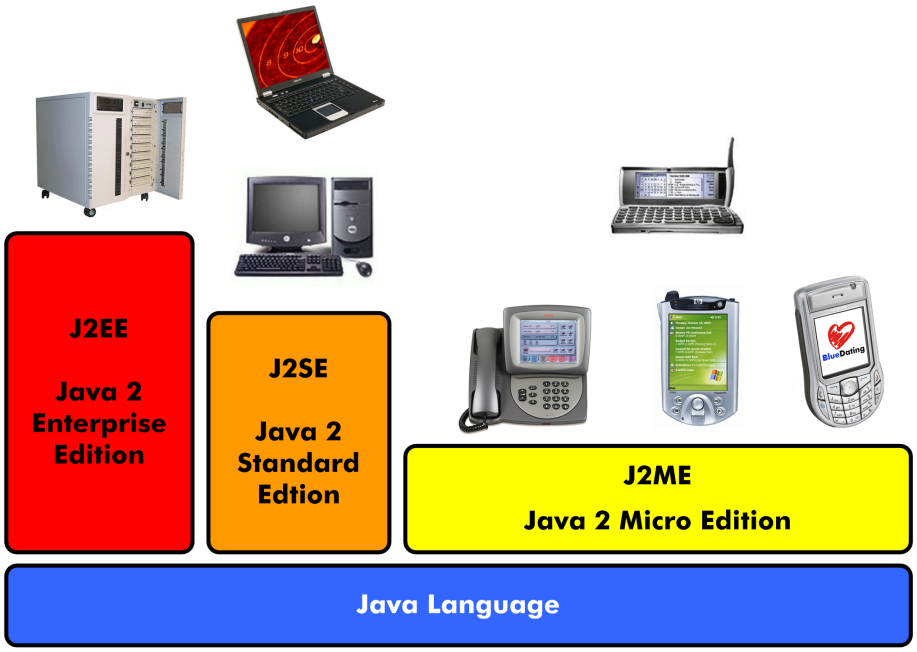


Abbildung 3.1: *Java 2 Plattformen*

3.1.1 J2ME Architektur

J2ME beschreibt ein Konzept von Konfigurationen und Profilen, welche auf die Java-Plattform angepasst werden.

Die Abbildung 3.2 zeigt den Aufbau der J2ME-Architektur. Sie besteht im wesentlichen aus folgenden Teilen:

- Das Betriebssystem
- Die virtuelle Maschine
- Die Konfiguration
- Die Profile

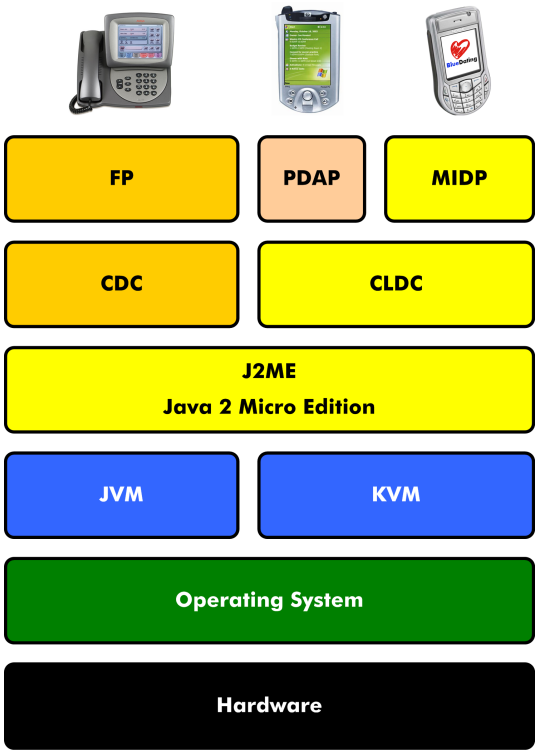


Abbildung 3.2: *Die J2ME Architektur*

3.1.1.1 Das Betriebssystem

Bei unseren Testgeräten von Nokia war das Betriebssystem das Symbian von Nokia. Es kann natürlich auch ein anderes Betriebssystem von einem anderen Hersteller sein, die Bedingung ist lediglich, dass es eine Schnittstelle für Java anbietet.

3.1.1.2 Die virtuelle Maschine

Unter J2ME gibt es zwei verschiedene virtuelle Maschinen. Die JVM (Java Virtual Machine) ist von J2SE her bekannt. JVM verifiziert den Bytecode zur Laufzeit, was auf eingebetteten System schlecht machbar wäre. Aus diesem Grund wurde die KVM (Kilobyte Virtual Machine) entwickelt. Die KVM bedient sich einem *Preverifier*, welcher nach dem Kompilieren dem Bytecode eine Prüfsumme beifügt. Zur Laufzeit muss danach nur noch diese Prüfsumme geprüft werden (mittels eines *runtime verifier*). Unter J2ME wird JVM meistens nur noch für Geräte genutzt, welche nicht mit Batterie betrieben werden, oder welche nicht allzu stark ressourcenlimitiert sind.

3.1.1.3 Die Konfiguration

Eine Konfiguration definiert die zur Verfügung stehenden APIs [1]. Unter J2ME gibt es zwei verschiedene Konfigurationen:

- Die *Connected Device Configuration* (CDC)
- Die *Connected Limited Device Configuration* (CLDC)

Connected Device Configuration (CDC)

CDC arbeitet mit der standard JVM, welche auch von J2SE genutzt wird. Diese Konfiguration ist demnach für Geräte mit mehreren Megabytes Speicher gedacht. Ebenso wird ein schneller Prozessor vorausgesetzt (32Bit). In Frage kommen somit vor allem fest installierte Geräte, wie z.B.: *Screen Phones*.

Connected Limited Device Configuration (CLDC)

Auf mobile Geräte zugeschnitten ist die CLDC, welche die KVM benutzt, um Ressourcen zu sparen. Für CLDC wird nur noch etwa 512KB Speicher vorausgesetzt. Ebenfalls reicht eine CPU mit 16 Bit. Netzverbindungen setzen keine hohen Datenraten voraus, sondern sind für kabellose Verbindungen optimiert. Um all dies zu erreichen, mussten die APIs der J2SE reichlich reduziert werden. Die Version 1.0 von CLDC unterstützt keine Gleitkommarechnung und die mathematischen Funktionen wurden aufs Minimum reduziert. Trotz all den Reduktionen besitzt CLDC die Möglichkeit, mehrere Threads zum gleichen Zeitpunkt auszuführen. Ebenfalls die von JAVA bekannte *Garbage Collection* blieb erhalten, um den Geräten freigegebenen Speicher möglichst schnell wieder zur Verfügung zu stellen. Die Version 1.1 von CLDC unterstützt neu jetzt auch Gleitkommarechnung, ist aber für kleine Geräte ungeeignet.

3.1.1.4 Die Profile

Da es verschiedenste Geräte gibt, die für CLDC geeignet sind, diese aber trotzdem sehr unterschiedliche Eigenschaften haben (Bildschirmgrösse, Bedienbarkeit...), wurden Profile für Geräteklassen entworfen. Die zwei wichtigsten Profile für CLDC sind:

- Mobile Information Device Profile (MIDP) für CLDC
- Personal Digital Assistant Profile (PDAP) für CLDC

Unter MIDP fallen auch mobile Telefone, wie sie für BlueDating gebraucht werden. MIDP wird im Abschnitt 3.2 detaillierter erläutet. PDAP ist für PDAs geeignet, welche ein Display hoher Auflösung und einen Stift als Eingabegerät aufweisen.
Ebenfalls für CDC gibt es diese Spezifizierung in Profile. Als Beispiel sei das *Foundation Profile (FP)* genannt, welches für Set-Top Boxen geeignet ist, die genügend Ressourcen für JVM und eine Netzwerkverbindung besitzen. Ein GUI ist nicht von nöten.

[1] *Application Program Interfaces* sind Bibliotheken mit Klassen, welche dem Entwickler zur Verfügung stehen

3.2 MIDP (Mobile Information Device Profile)

MIDP ist ein Profil für mobile Geräte, welche bestimmte Eigenschaften besitzen. Das Gerät muss ein Display von mindestens 96 auf 54 Pixel haben. Ebenfalls muss es mindestens 256kB nichflüchtigen Speicher und mindestens 32kB flüchtigen Speicher besitzen um den MIDP 2.0 Anforderungen zu entsprechen. Auf der Software Seite muss das Gerät die KVM laufen lassen können. MIDP stellt dem Entwickler ein Framework zur Ausführen von MIDlets[2] zur Verfügung.

3.2.1 Das MIDlet

Es können mehrere MIDlets in einer MIDlet-Suite zusammengefasst werden. Alle MIDlets laufen unter dem MIDP-Profil.
Ein MIDlet hat 3 Lebenszustände:

- **Paused**: Ein MIDlet befindet sich in diesem Zustand, wenn es gerade nicht genutzt wird.
- **Active**: Ein MIDlet befindet sich in diesem Zustand, wenn es gerade ausgeführt wird.
- **Destroyed**: Dieser Zustand zeigt, dass das MIDlet alle genutzten Ressourcen freigegeben hat und der Garbage Collector den Speicher aufräumen kann.

Um ein MIDlet zu entwickeln muss man mindestens 3 Methoden implementieren, welche den obig genannten Lebenszuständen entsprechen. Das MIDlet enthält dagegen keine *main* Methode. Anstelle dieser tritt die *startApp()* Methode. Diese 3 Methoden müssen unter folgenden Namen definiert sein:

- abstract void **startApp**()
- abstract void **pauseApp**()
- abstract void **destroyApp**(boolean unconditional)

3.2.2 Die Benutzeroberfläche

Die *displayable* Klasse beschreibt alle möglichen Objekte, welche auf dem Display dargestellt werden können. Die Abbildung 3.3 zeigt dessen Klassenhierarchie.
Die Benutzeroberfläche kann in zwei Arten beschrieben werden. Entweder mittels *High-Level APIs* oder mittels *Low-Level APIs*.

3.2.2.1 High-Level APIs

Diese Art, das Display zu beschreiben, ist auf alle MIDP-kompatiblen Geräte portierbar. Da dies der beste Weg ist, menubasierte Applikationen zu entwickeln, ist BlueDating mit High-Level APIs entwickelt worden.
Die *screen* Klasse in Abbildung 3.3 beschreibt die High-Level APIs. Die wichtigsten High-Level APIs sind:

[2]Ein MIDlet ist eine MIDP-Anwendung, welche nur die APIs aus MIDP und CLDC benutzt. Vergleichbar mit einem Java-Applet

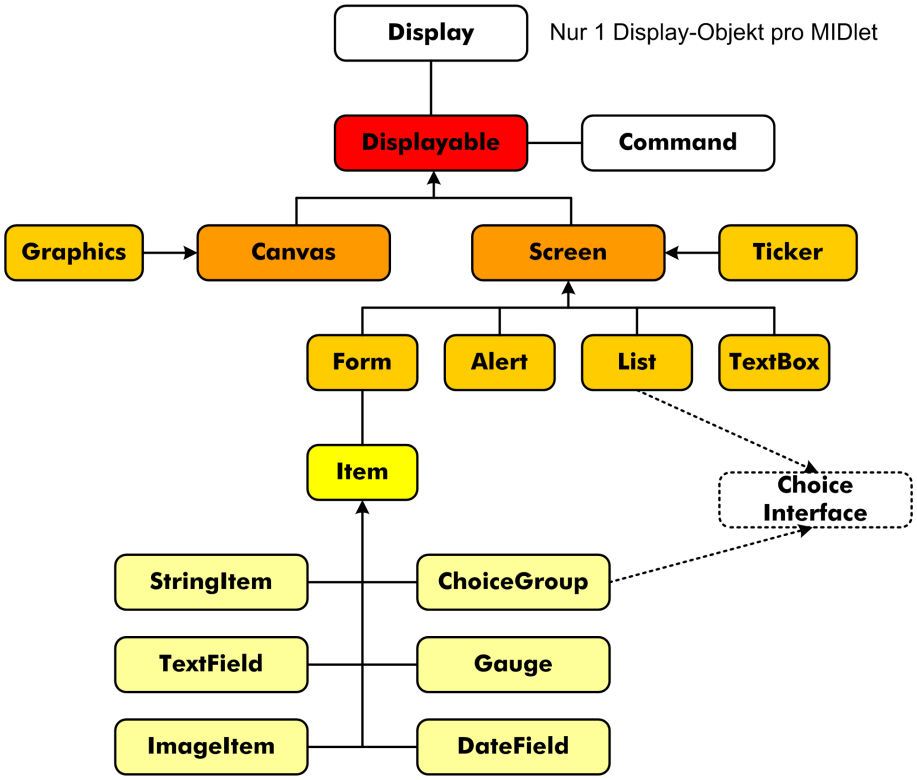


Abbildung 3.3: *Die Klassenhierarchie der Benutzeroberflächen-Klassen*

- **Ticker**: Eine Laufschrift im oberen Bereich des Displays.
- **TextBox**: Zeigt eine Texteingabefeld an.
- **List**: Zeigt eine Liste an.
- **Alert**: Gibt die Möglichkeit Alarm-Popup auszugeben
- **Form**: Ermöglicht die Darstellung mehrerer Komponenten in einem Fenster.

Innerhalb einer Form können folgende Objekte eingefügt werden:

- **StringItem**: Für statischen Text.
- **TextField**: Für die Text Eingabe.
- **ImageItem**: Um Bilder anzuzeigen.
- **ChoiceGroup**: Um Auswahllisten darzustellen.
- **Gauge**: Um einen Fortschrittsbalken anzuzeigen.
- **DateField**: Um Datum und Zeit anzuzeigen.

Die einzelnen Objekte in einer Form werden vom Mobilgerät dann automatisch auf dem Bildschirm angeordnet.
Über die *Command* Klasse werden Befehle des Benutzers registriert. Für ein displayable muss dann ein *CommandListener* definiert sein: Eine Funktion, die ausgeführt wird, wenn der Benutzer einen Befehl eingibt und entsprechend auf diesen reagiert.

3.2.2.2 Low-Level APIs

Diese APIs haben den Vorteil, das Display genau zu beschreiben. Die Klasse *Canvas* in Abbildung 3.3 beschreibt diese APIs. Für Spiele ist dies die geeigneste Variante Grafik schön darzustellen. Man muss jedoch den Kompromiss eingehen, dass die Applikation damit eventuell nicht mehr auf alle MIDP-Geräte portierbar ist.

3.2.3 Die JAR- und JAD-Datei

JAR-Archiv

Ein MIDlet wird zur Benutzung auf dem mobilen Gerät in einem komprimierten Archiv zusammengefasst, welches alle kompilierten Programmdateien und Bilder enthält. Die Dateiendung solcher Archive lautet auf *jar*, was soviel wie *JAVA Archiv* bedeutet. Innerhalb des Archivs wird die Applikation durch eine Manifest-Datei beschrieben, welche die wichtigsten Eigenschaften der Applikation beschreibt.

JAD-Datei

Um jedoch alle Informationen über die Applikation bereits vor deren Installation oder Dekomprimierung zu erhalten, bedient man sich einer *Java-Application Descriptor*-Datei, welche im Prinzip nochmals dieselben Informationen des Manifests innerhalb des Archivs besitzt. Diese Datei wird separat zum Archiv mitgeliefert und kann mit den meisten Entwicklertools direkt erstellt werden.

Die JAD-Datei muss folgende Attribute enthalten:

Attribut	Beschreibung
MIDlet-Version	Versionsnummer des MIDlets
MIDlet-Vendor	Hersteller
MIDlet-Jar-URL	URL zum Download
MicroEdition-Configuration	Name und Version der Konfiguration
MicroEdition-Profile	Name und Version des Profils
MIDlet-Jar-Size	Grösse des JAR-Archivs
MIDlet-Name	Name des MIDlet
MIDlet-1	Name, Icon und Klasse des MIDlets

Tabelle 3.1: *Attribute in der JAD-Datei*

Unsere BlueDating.jad Datei sieht demnach folgenermassen aus:

```
MIDlet-Version              1.0
MIDlet-Vendor               Sandro Schifferle, Christian Braun
MIDlet-Jar-URL              BlueDating.jar
MicroEdition-Configuration  CLDC-1.0
MicroEdition-Profile        MIDP-2.0
MIDlet-Jar-Size             81530
MIDlet-Name                 BlueDating
MIDlet-1                    BlueDating, /icon.png, BlueDating
```

3.3 Record stores

MIDP bietet zur permanenten Speicherung von Daten das *Record Management System* (RMS) an. Damit können Daten als Byte Arrays in sogenannten *Records* abgelegt werden, wobei mehrere Records in *RecordStores*, welche über einen Namen identifizierbar sind, zusammengefasst werden. Die einzelnen Records können über eine im RecordStore eindeutige *Record ID* zugegriffen werden, die beim Erstellen neuer Records automatisch zugewiesen wird.
Da die RecordStore-Spezifikation dem Plattformhersteller Freiheiten lässt, was für Record IDs zugewiesen werden (sie müssen lediglich eindeutig sein innerhalb des RecordStores), kann nicht davon ausgegangen werden, dass die IDs von zugefügten Records bei 1 beginnend sequentiell ansteigen, wie dies bei Sun's Referenzimplementation der Fall ist. [10]
Die RecordStores können jeweils von dem MIDlet gelesen werden, das sie erstellt hat, sowie von anderen MIDlets derselben Suite. Entsprechend müssen die RecordStore-Namen innerhalb einer MIDlet Suite eindeutig sein.

3.4 Dateisystem Zugriff

Der Zugriff auf das Dateisystem des Mobilgeräts ist über das *FileConnection API*, welches in *JSR-75* [11] spezifiziert ist, möglich. Die Verbreitung dieses APIs ist auf heutigen Mobiltelefonen noch nicht allzu gross, weshalb wir darauf verzichtet haben, dessen Funktionen in unserem Programm BlueDating zu nutzen.

Kapitel 4

Bluetooth

4.1 Einleitung

Bluetooth ist ein Protokoll für drahtlose Kommunikation. Es besitzt wie viele andere Kommunikationsprotokolle eine Server-Client Struktur. Bluetooth funkt auf 2.4 GHz, also der gleichen Frequenz wie "Wireless Lan 802.11b". Die beiden Technologien beeinflussen sich jedoch gegenseitig nicht. Es gibt drei verschiedene Geräteklassen, welche in der Sendestärke unterschieden werden (siehe Tabelle 4.1).

Klasse	Sendestärke	Reichweite
Class 1	100 mW	100 Meter
Class 2	2.5 mW	20 Meter
Class 3	1 mW	10 Meter

Tabelle 4.1: *Bluetooth Geräteklassen*

Die Bluetooth Technologie ist also gedacht, um Geräte im nahen Bereich drahtlos zu verbinden. Die Übertragungsgeschwindigkeit ist mit 1 Mb/s im Vergleich zu 11 Mb/s bei 802.11b eher klein. Bluetooth ist auch nicht gedacht um grosse Datenmengen hin und her zu kopieren, sondern um eine einfache Kommunikation zwischen Geräten herzustellen, wie es für mobile Geräte nötig ist. Es kann auch als Ersatz für die Serielle Schnittstelle RS-232 oder USB genutzt werden, um Geräte drahtlos und schnell anzubinden. Heutzutage besitzen schon viele mobile Geräte Bluetooth, was man meistens am aufgedruckten Bluetooth-Logo erkennen kann (siehe Abbildung 4.1).

Abbildung 4.1: *Das Bluetooth Logo*

Ältere Geräte können auch mit Bluetooth nachgerüstet werden. Abbildung 4.2 zeigt eine SD Karte für Palm OS 4 Geräte.

Der Bluetooth Protocol Stack

Um einem Gerät die Fähigkeit zu geben mit anderen Geräten über Bluetooth zu kommunizieren, braucht es einen einheitlichen Protocol Stack, um dies der grossen

Abbildung 4.2: *Das Palm SD Bluetooh Modul*

Gerätevielfalt zur Verfügung zu stellen. Dies ist der *Bluetooth Protocol Stack*. Die Abbildung 4.3 zeigt den *Bluetooth Protocol Stack*. In der Tabelle 4.2 sind die einzelnen Layer detaillierter bezeichnet. Einzelne Layer werden ebenfalls als Protokoll bezeichnet, weil es sich um "Unterprotokolle" vom *Bluetooth Protocol Stack* handelt.

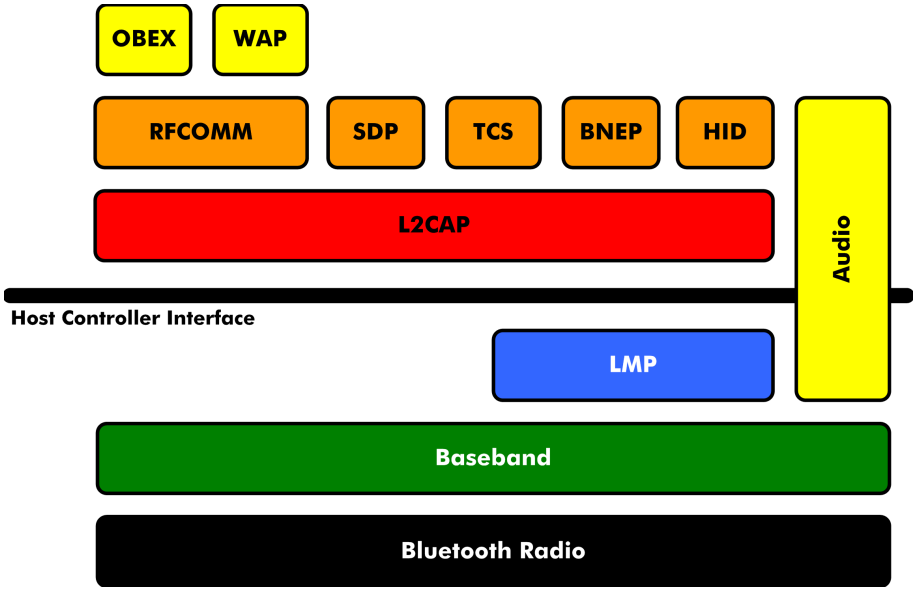


Abbildung 4.3: *Der Bluetooth Protokoll Stack*

Layer	Beschreibung
Bluetooth Radio	Dieser Layer definiert die Anforderungen an den Bluetooth Transceiver.
Baseband	Dies ist der physikalische Layer von Bluetooth. Hier werden physikalische Kanäle und Verbindungen zu anderen Geräten verwaltet.
LMP	Das Link Manager Protocol wird von den Link Managern für den Linkaufbau, die Linkkontrolle und Linkkonfiguration genutzt.
Host Controller Interface	Dieser Layer beschreibt das Interface über welches auf die Hardware zugegriffen werden kann.
Audio	Dies wird zur Audioübertragung genutzt.
L2CAP	Das Logical Link Control and Adaptation Protocol unterstützt das Multiplexen von Protokollen aus höheren Levels.
RFCOMM	Das RFCOMM Protokol bietet das Emulieren eines seriellen Anschlusses über das L2CAP Protokol an.
SDP	Das Service Discovery Protocol stellt die Suche nach anderen Bluetooth Geräten und deren Services für die Applikationen zur Verfügung.
TCS	Die Telephony Control Protocol Specification stellt Audio Schnittstellen zur Verfügung.
BNEP	Das Bluetooth Network Encapsulation Protocol ist ein Layer, der es ermöglicht andere Netzwerk Protokolle in L2CAP Pakete zu verpacken.
HID	Das Human Interface Device Protocol stellt Schnittstellen zu Eingabegeräten zur Verfügung
OBEX	Der Layer Object Exchange ermöglicht das Senden und Empfangen von Objekten.
WAP	Das Wireless Access Protocol gestattet das Betrachten von WAP-Seiten.

Tabelle 4.2: *Beschreibung der Bluetooth Layer*

Profile

Um alle Geräte, welche Bluetooth benutzen, zu klassifizieren, wurden sogenannte Bluetooth Profile erstellt, die zum Teil voneinander abhängig sind. Dies ist sinnvoll, da somit nicht alle Geräte alles können müssen. Die Abbildung 4.4 zeigt eine Übersicht der wichtigsten Profile. Auf `www.bluetooth.org` findet sich eine vollständige Auflistung.
Die Namen der Profile sind meistens selbst aufschlussreich darüber, um welche Art von Produkt es sich handelt. Der Verschachtelung wegen muss zum Beispiel das *FAX Profile* ebenfalls das *Serial Port Profile* und das *Generic Access Profile* unterstützen.

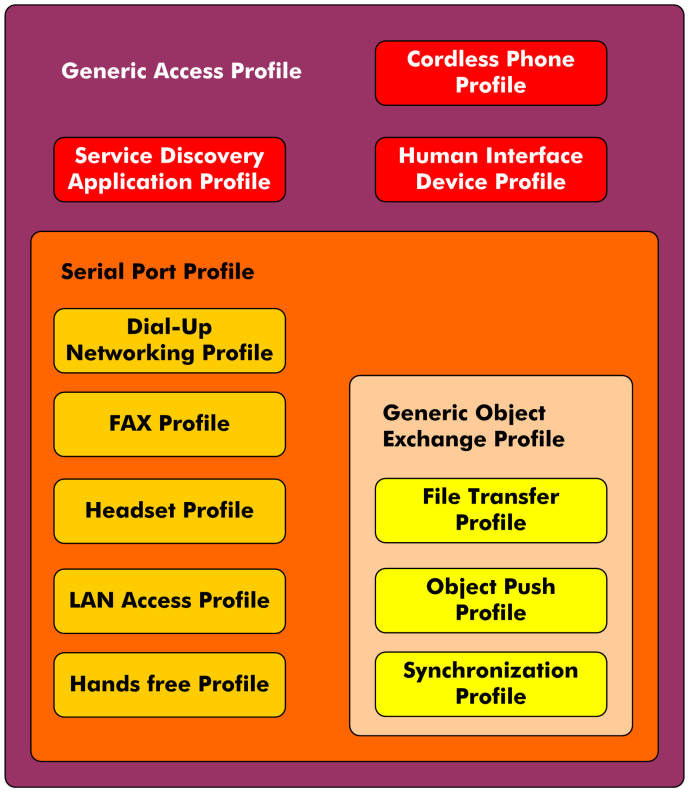


Abbildung 4.4: *Die Bluetooth Profile*

4.2 Bluetooth mit J2ME

Um ein Bluetooth-Gerät mit Java zu steuern, muss dieses JSR-82 [4] unterstützen. JSR-82 ist der Name für das Bluetooth Java API. Das Java API ist unabhängig von der verwendeten Bluetooth Hardware, was ihm einen grossen Vorteil gegenüber APIs in hardwarenahem C verschafft. Jede Hardware, die diesen Standard unterstützen will, muss mindestens folgende Layer des Bluetooth Stacks besitzen (Es werden auch nur diese unterstützt):

- Host Controller Interface (HCI)
- Logical Link Control and Adaption Protocol (L2CAP)
- Service Discovery Protocol (SDP)
- RFCOMM

Ebenfalls müssen folgende Bluetooth Profile unterstützt werden:

- Generic Access Profile
- Service Discovery Application Profile
- Serial Port Profile
- Generic Object Exchange Profile

Braucht man andere Layer oder andere Profile, kann man Java nicht dafür benutzen. Zum Beispiel müsste man eine Distanzmessung mittels Signalstärke in C implementieren.
In J2ME kann mittels den Java APIs `javax.bluetooth` und `javax.obex` auf Bluetooth zugegriffen werden.
Da Bluetooth eine Server-Client Struktur besitzt, muss zuerst ein Server definiert werden, damit man mit einem Client einen Server suchen und mit diesem kommunizieren kann.

4.2.1 Server einrichten

Um dem Server eine Kennung zu geben, wird in Bluetooth etwas Ähnliches wie eine IP-Adresse vergeben. In Bluetooth wird dies *URL* gennant. Eine *URL* hat normalerweise folgende Struktur:

```
btspp://localhost:FF197E65C67A348BB752C973C465D15F;
name=Bluedate Server;authorize=false
```

Wobei `btspp://localhost:` bezeichnet, dass es sich um eine RFCOMM Verbindung handelt, und zwar mit dem *Bluetooth Serial Port Profile*. Die nachfolgende 32 Bit Zahl ist der UUID (*Universal Unique Identifier*). Der UUID ist eine einmalige Nummer, welche den Service beschreibt, der auf dem Bluetooth Gerät angeboten werden soll. Eine Liste der schon vergebenen UUID Nummern kann unter [5] eingesehen werden.
Um den Serverprozess mit dieser *URL* zu öffnen, bedient man sich folgenden Aufrufs:

```
StreamConnectionNotifier notifier = (StreamConnectionNotifier)
Connector.open(url);
```

Mit diesem Befehl wird ein *Notifier* für diese *URL* erstellt. Um nun eingehende Clientanfragen zu beantworten können, muss man dem *Notifier* mitteilen, was er mit der Anfrage anstellen soll. Dazu verwendet man den folgenden Aufruf:

```
StreamConnection conn = notifier.acceptAndOpen();
```

Dieser Aufruf legt den aufrufenden Thread solange schlafen, bis ein Client antwortet und gibt dann eine *StreamConnection* zurück, welche die Referenz zum Client enthält.

4.2.2 Geräte- und Servicesuche

Um als Client mit einem Server über Bluetooth zu kommunizieren, muss er zuerst alle Bluetooth-Geräte im Umkreis finden und deren angebotenen Services abfragen. Danach vergleicht der Client die gefundenen UUIDs mit der UUID von der er einen Server sucht.
Um eine Suche nach umliegenden Geräten mit aktiviertem Bluetooth zu starten benutzt man folgende Methode:

```
javax.bluetooth.DiscoveryAgent.startInquiry();
```

Diese Methode ruft für jedes gefundene Gerät die Methode `deviceDiscovered()` und nach beendeter Suche die Methode `inquiryCompleted()` auf.
Um ein Gerät nach vorhandenen Services abzufragen benutzt man folgende Methode:

```
javax.bluetooth.DiscoveryAgent.searchService();
```

Diese Methode nutzt das *Service Discovery Application Profile* um über das *Service Discovery Protocol (SDF)* mit anderen Geräten zu kommunizieren. Diese Methode ruft für jedes gefundene Gerät, welches den gesuchten Service anbietet, die Methode `servicesDiscovered()` und nach beendeter Suche die Methode `serviceSearchCompleted()` auf.
Mann kann nun durch die erhaltene *URL* vom Server mit folgendem Befehl eine Verbindung zum Server öffnen:

```
StreamConnection conn = (StreamConnection)Connector.open(url);
```

Die somit erhaltene *StreamConnection* ist die Referenz zum verbundenen Server, der jetzt ebenfalls eine *StreamConnection* zu diesem Client erhalten hat.

4.2.3 Kommunikation

Da RFCOMM "streamorientiert" ist, muss man nur noch einen Lese- und Schreibstream zum Gegenüber öffnen. Dies erfolgt mit folgenden Aufrufen:

```
OutputStream out = conn.openOutputStream();
InputStream in = conn.openInputStream();
```

Man kann nun mit der Methode `read()` in einen Stream schreiben und mit der Methode `write()` von einem Stream lesen. Die `read()` Methode legt den Thread so lange schlafen, bis neue Daten anliegen.
Neben dem "streamorientierten" RFCOMM ist in Java ebenfalls eine Verbindung über das "paketorientierte" L2CAP möglich. Die standard MTU (Maximum Transmission Unit) ist auf 672 bytes gesetzt. Diese kann nicht ohne Kompatibilitätsprobleme einfach so erhöht werden. Da einzelne "Pakete" von BlueDating, grösser als diese default MTU sind, benutzt BlueDating die RFCOMM Übertragung.

4.3 Erfahrungen mit dem Nokia 6630

Das wichtigste ist natürlich die Reichweite, bei der *BlueDating* noch funktionert. Die Abbildung 4.5 veranschaulicht die im Test erreichten Distanzen. Für die Messung mit Hindernis begaben wir uns mit einem Gerät in den Nebenraum.
Ein Verbindungsaufbau braucht auch seine Zeit. Die Geräte- und Servicesuche, wie sie im Abschnitt 4.2.2 beschrieben ist, kann über Java nicht beeinflusst werden. Durchschnittlich dauerte diese Suche 14 Sekunden. Die komplette Kommunikation der BlueDating Software dauert nach dieser Suche nochmals etwa 4 Sekunden. Eine Suche nach möglichen Partnern geht also rund 18 Sekunden.
Leider ist das JSR-82, welches im Nokia 6630 implementiert ist, noch fehlerhaft und sehr instabil. Sobald mehrere Geräte eine Geräte- oder Servicesuche gleichzeitig durchführen, wird oft gar kein Gerät gefunden und in schlimmen Fällen kann es

Abbildung 4.5: *Die maximale Einsatzdistanz von BlueDating mit dem Nokia 6630*

gar zu Abstürzen kommen. Der Grund ist eventuell die Einschränkung, dass immer nur eine *session* zwischen zwei Geräten vorhanden sein kann, was zu Komplikationen führen könnte, wenn beide gleichzeitig versuchen eine Verbindung aufzubauen. Diese Kollisionsgefahr erhöht sich natürlich mit der Anzahl Geräte in Reichweite. Unser Test mit 4 Geräten zeigte diese Schwächen klar auf. Abhilfe könnte beispielsweise ein verlängertes Suchintervall schaffen, da sich dadurch seltener mehrere Geräte im kritischen Bereich der Geräte- und Servicesuche befinden.
Ein zweites Problem besteht mit den Input- und Outputstreams. Schliesst man einen solchen Stream (mittels `stream.close()`), kann zum gleichen Partner keine neue Verbindung mehr aufgebaut werden, obwohl man die *StreamConnection* zum anderen Gerät noch besitzt. Eine Abhilfe ist ein neues Entdecken des Gerätes, was ja nicht im Sinne eines Kommunikationsablaufs sein kann und lange dauert. Oder man öffnet den Stream am Anfang einer Kommunikation und schliesst ihn erst, nachdem alle Daten geflossen sind. Diese zweite Variante ist in BlueDating implementiert.
Als Rettung für diese Probleme ist ein baldiges Firmware Update für das Nokia 6630 zu erhoffen. Denn im Emulator von Nokia treten diese "Phänomene" nicht auf (Siehe auch unter [13]).

Kapitel 5

Konzept

5.1 Allgemeines

Dieses Kapitel soll erläutern, was ein Benutzer der BlueDating-Software alles erwarten kann und wie er erreichen kann, was er will.
Wie eingangs erwähnt, dient BlueDating dazu, andere Menschen zu finden, deren Eigenschaften mit gewissen Wunschvorstellungen übereinstimmen. Die Applikation BlueDating ist, wie dem Namen zu entnehmen ist, primär für die Partnersuche ausgelegt, sowie für die Suche von Freundschaften. Ein Profil des Benutzers wird dabei mit der Suchanfrage eines anderen Geräts verglichen, ist eine Übereinstimmung vorhanden, wird dies dem Benutzer mitgeteilt, es entsteht ein "Match".
Grundsätzlich lässt sich aber das Konzept an diverse andere Situationen anpassen, in welchen es von Interesse ist, Informationen zu Personen im näheren Umkreis zu erlangen und diese aufgrund gewisser Suchkriterien auszuwerten. So könnten beispielsweise Kongressbesucher ihre Interessen angeben, so dass sie dann benachrichtigt werden, wenn sie einer Person über den Weg laufen, welche eine Fachkraft auf dem gesuchten Gebiet ist.
Das Konzept liesse sich sogar insofern erweitern, dass nicht zwingend alle beteiligten Geräte von Menschen getragen werden. Beispielsweise liesse sich ein Programm zur Suche von passenden Hotels programmieren: Der Benutzer erstellt auf seinem Mobilgerät ein Profil mit seinen Wünschen (Doppelzimmer mit Balkon, Preisklasse, Anzahl Sterne...). Das Gerät kommuniziert dann mit fest installierten Bluetooth-Geräten, welche sich beim Hotel befinden. Diese vergleichen die Suchanfrage mit den freien Zimmern, die sie anbieten können und melden das Resultat zurück. Gibt es passende Zimmer, wird dies dem Benutzer angezeigt. Der Vorteil dieser Lösung ist, dass sich der Benutzer zum Zeitpunkt der Meldung gerade vor dem betreffenden Hotel befindet.
All diese möglichen Anwendungen haben folgendes gemeinsam: Information soll mit anderen Geräten ausgetauscht werden, wobei durch die definierten Suchkriterien eine gewisse Evaluierung stattfindet, um den Benutzer nur mit einer Meldung zu belästigen, wenn das Gefundene seinem Interesse entspricht. Dabei ist die örtliche Nähe der beiden Geräte wichtig. So ist sichergestellt, dass das Gesuchte nicht nur irgendwo vorhanden ist, sondern sich in der unmittelbaren Umgebung befindet.
Die relativ geringe Reichweite der Datenfunktechnik Bluetooth legt die Grenze für diese nähere Umgebung fest. Eine andere Möglichkeit besteht zwar darin, das Gerät zu orten und die Position an einen Server zu übertragen, der mit Informationen zur Umgebung aufwartet. Doch das Bluetooth-System hat den Vorteil, dass es wegen seines dezentralen Charakters einen viel geringeren Aufwand erfordert.

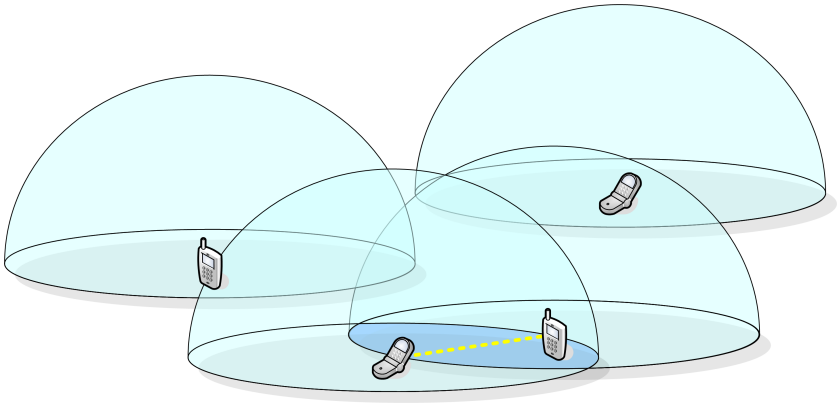


Abbildung 5.1: *Zwei Geräte können über Bluetooth kommunizieren, wenn sie nahe genug beieinander sind.*

5.1.1 Kommerzielle Variante

Auch wenn die von uns implementierte Version auf kommerzielle Elemente verzichtet, ist es vom Konzept her grundsätzlich möglich, solche einzubauen. Weitere Details dazu finden sich im Abschnitt 5.3.3.

5.2 Profil

Damit BlueDating seine Hilfe bei der Partnersuche leisten kann, benötigt es Informationen einerseits über die Eigenschaften des Benutzers und andererseits über die Eigenschaften, die der Benutzer von seinem Partner erwartet. Diese Angaben bilden das Profil des Benutzers.

5.2.1 Teile des Profils

Das Benutzerprofil umfasst fünf Teile, welche sich in zwei Gruppen gliedern: Die Gruppe der Angaben über den Benutzer (oft kurz als "eigene" Angaben bezeichnet), welche den Benutzer charakterisieren, sowie die Gruppe der Suchangaben ("gesuchte" Angaben), welche beschreiben, was für einen Partner der Benutzer sucht.
Die Angaben über den Benutzer umfassen die Teile Basisangaben, erweiterte Angaben sowie zusätzliche Informationen. Die Gruppe der Suchangaben wird durch die Basisangaben und die erweiterten Angaben gebildet.

5.2.1.1 Eigene Basisangaben

Die eigenen Basisangaben sind in der Tabelle 5.1 aufgelistet. Es handelt sich dabei um Äusserlichkeiten und grundlegende Eigenschaften. Es lässt sich jeweils nur genau ein Wert bzw. eine Eigenschaft wählen.
Die eigenen Basisangaben sind zwingend anzugeben, beim ersten Programmstart wird direkt die Eingabemaske dafür angezeigt.

5.2.1.2 Eigene erweiterte Angaben

Die eigenen erweiterten Angaben müssen nicht zwingend eingegeben werden. Es sind gegensätzliche Wortpaare, mit denen sich der Benutzer genauer charakterisieren

Eigenschaft	Einheit	Werte
Geschlecht		[männlich, weiblich]
Alter	Jahre	[18 .. 99]
Körpergrösse	cm	[100 .. 250]
Gewicht	kg	[40 .. 250]
Haarfarbe		[blond, rot, braun, schwarz, grau, gefärbt, Glatze]
Augenfarbe		[grün, blau, braun]
Rauchgewohnheiten		[nie, manchmal, täglich]
Trinkgewohnheiten		[nie, manchmal, im Ausgang, täglich]

Tabelle 5.1: *Eigene Basisangaben*

kann, indem er auf einer Skala von 1 bis 5 wählt, welche der beiden Eigenschaften eher auf ihn zutrifft. Die Eigenschaften sind in Tabelle 5.2 aufgelistet.

Wort 1	Skala	Wort 2
Bewegungsfaul	[1 .. 5]	Sportfanatiker
Lesen	[1 .. 5]	Kino/TV
Party/Ausgang	[1 .. 5]	Ruhiger Abend
Trampen	[1 .. 5]	5-Sterne-Hotel
Nachtmensch	[1 .. 5]	Früh ins Bett
Frühaufsteher	[1 .. 5]	Morgenmuffel
Harmonie	[1 .. 5]	Energie
Armani	[1 .. 5]	2nd Hand
Papier und Bleistift	[1 .. 5]	Computerfreak
Take Out/Fast Food	[1 .. 5]	Restaurant
Weltentdecker	[1 .. 5]	Stubenhocker
Stadtmensch	[1 .. 5]	Landmensch
Kein Haustier	[1 .. 5]	"Zoo" daheim
Indoor	[1 .. 5]	Outdoor
Einzelgänger	[1 .. 5]	Gruppenmensch

Tabelle 5.2: *Erweiterte Angaben*

5.2.1.3 Zusätzliche Information

Diese Daten werden, wenn ein Match zustande gekommen ist, automatisch auf das andere Mobilgerät übertragen. Es handelt sich um Kontaktinformationen sowie um ein Feld "Kommentar", in welchem ein frei wählbarer Text eingegeben werden kann (siehe Tabelle 5.3).

5.2.1.4 Gesuchte Basisangaben

Die gesuchten Basisangaben, welche der Tabelle 5.4 zu entnehmen sind, entsprechen weitgehend den eigenen Basisangaben. Der Unterschied ist einerseits, dass sich mehrere Eigenschaften auswählen lassen (bzw. für numerische Werte eine Ober- und

Eingabefeld	Max. Länge	Form
Telefonnummer	20	Nur Zahlen
E-Mail-Adresse	50	@ muss vorhanden sein
Kommentar	500	Freier Text

Tabelle 5.3: *Zusätzliche Informationen*

eine Untergrenze angegeben werden kann), andererseits kommt noch die Eigenschaft "Beziehungsart" hinzu.

Eigenschaft	Einheit	Werte
Geschlecht		[männlich, weiblich]
Alter von	Jahre	[18 .. 99]
Alter bis	Jahre	[18 .. 99]
Körpergrösse von	cm	[100 .. 250]
Körpergrösse bis	cm	[100 .. 250]
Gewicht von	kg	[40 .. 250]
Gewicht bis	kg	[40 .. 250]
Haarfarbe		[blond, rot, braun, schwarz, grau, gefärbt, Glatze]
Augenfarbe		[grün, blau, braun]
Rauchgewohnheiten		[nie, manchmal, täglich]
Trinkgewohnheiten		[nie, manchmal, im Ausgang, täglich]
Beziehungsart		[Sex, Freundschaft, lockere Beziehung, enge Beziehung]

Tabelle 5.4: *Gesuchte Basisangaben*

5.2.1.5 Gesuchte erweiterte Angaben

Die gesuchten erweiterten Angaben sind identisch zu den eigenen erweiterten Angaben, welche aus Tabelle 5.2 hervorgehen.

5.3 Matching

Wenn sich zwei BlueDater begegnen, müssen die beiden Profile mit den im Abschnitt 5.2.1 beschriebenen Daten miteinander verglichen werden. Das eine Profil muss den Suchkriterien des anderen Profils entsprechen und umgekehrt. Dies geschieht in zwei Stufen, zuerst mit den Basisangaben, stimmen diese überein werden anschliessend die erweiterten Angaben verglichen. Abbildung 5.2 illustriert diesen Vorgang. Dieser ganze Prozess stellt das "Matching" dar. Ist dieses erfolgreich, also ein Match erreicht, tauschen die Geräte die zusätzlichen Informationen aus.

5.3.1 Matching der Basisangaben

Die Basisangaben müssen die Suchkriterien vollständig erfüllen, damit ein Match möglich ist. Der Benutzer erwartet, dass seine hier gesetzten Grenzen genau einge-

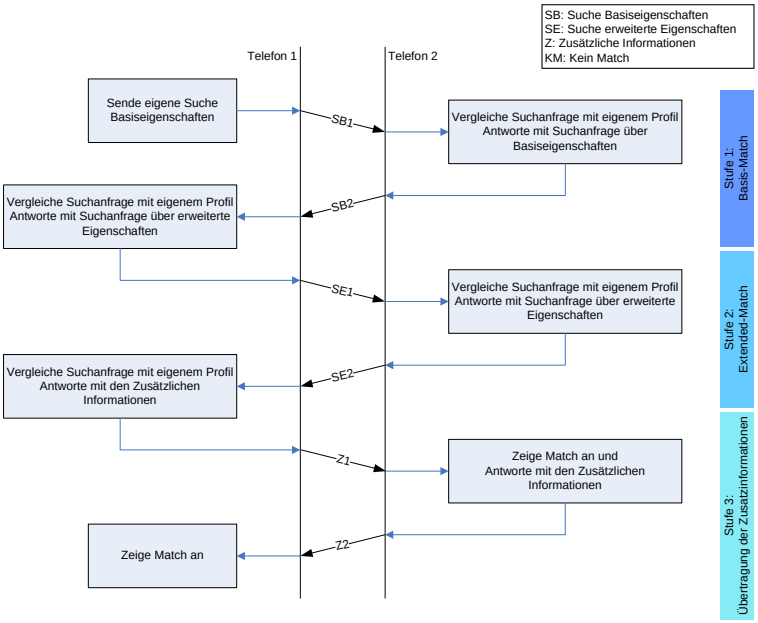


Abbildung 5.2: *Grober Ablauf des Matchings, wenn ein Match zustande kommt.*

halten werden. Wo der Benutzer keine Angabe im Suchprofil macht, ist ihm diese Eigenschaft egal.
Ein Match ist also möglich, wenn folgendes gegeben ist:

- Alle numerischen Werte des eigenen Profils sind innerhalb der im Suchprofil festgelegten Schranken. Es ist auch möglich, dass nur eine Ober- oder Untergrenze gegeben ist.

- Bei allen anderen Angaben ist die im eigenen Profil gewählte Eigenschaft in der Suchanfrage enthalten oder die Suchanfrage für die entsprechende Eigenschaft ist leer (keine Eigenschaft ausgewählt), was heisst, dass es dem Suchenden egal ist.

Sind diese Kriterien in beide Richtungen erfüllt (Suchprofile beider Geräte verglichen mit den eigenen Profilen des jeweils anderen Geräts), so ist ein Match für die Basisangaben erreicht und die erweiterten Angaben werden verglichen.

5.3.2 Matching der erweiterten Angaben

Die eigenen erweiterten Angaben müssen nicht exakt mit den gesuchten erweiterten Angaben übereinstimmen, dass ein Match erreicht wird. Die Abweichung soll einfach nicht zu gross sein, genaugenommen darf die durchschnittliche Abweichung pro Frage einen Schwellenwert nicht überschreiten. Natürlich muss diese Bedingung für einen Match wieder in beiden Richtungen erfüllt sein, also das eigene Profil des einen Gerätes muss der Suchanfrage des anderen genügen und umgekehrt.
Es besteht jedoch noch das Problem, dass die erweiterten Angaben freiwillig sind. Einerseits wird die Suchanfrage oft unvollständig sein, andererseits ist auch das eigene Profil lückenbehaftet. Fehlt eine einzelne Angabe in der Suchanfrage, ist sie offenbar für den Suchenden nicht von Interesse, entsprechend wird diese Frage nicht

ausgewertet. Der Durchschnittswert der Abweichungen wird nur für die Fragen berechnet, welche in der Suchanfrage vorkommen. Ist die Suchanfrage leer, so ist die Bedingung für einen Match wenigstens in dieser Richtung automatisch erfüllt.
Fehlt jedoch eine Angabe, die in der Suchanfrage vorhanden ist, im eigenen Profil, wird die Frage mit einer "Straf"-Abweichung von 4 Differenzpunkten gewertet. Dies, weil offenbar Differenzen bestehen, in der Gewichtung der Wichtigkeit der betreffenden Frage: Dem Suchenden ist sie offenbar wichtig, sonst hätte er sie in der Suchanfrage ausgelassen, dem Gesuchten scheint sie unwichtig, sonst hätte er die Frage beantwortet.

5.3.3 Matching bei kommerzieller Variante

Der Vorgang des Matchings, wie er in den vorigen Abschnitten beschrieben ist, läuft alleine zwischen jeweils zwei über Bluetooth verbundenen Geräten ab. Kommen kommerzielle Interessen ins Spiel, muss der Ablauf des Matchings an einer Stelle durch einen Schritt unterbrochen werden, der eine Verrechnung des Dienstes ermöglicht.
Vorstellbar wäre beispielsweise eine Variante, bei der die Geräte das Matching wie in der kostenfreien Version durchführen, jedoch wenn ein Match zustandekommt, diesen nicht direkt anzeigen, sondern den Benutzer auffordern, diesen kostenpflichtig freizuschalten. Der Abschnitt 6.1.2.3 zeigt entsprechende Möglichkeiten auf.

5.4 Match-Verwaltung

Wenn ein Match zustandegekommen ist, wird er auf dem Gerät in einer Liste mit bisherigen Matches gespeichert (siehe auch Kapitel 6.4). Die übertragenen Informationen sowie der Zeitpunkt des Matches und der Bluetooth-Name des anderen Gerätes lassen sich in dieser Liste detailliert anschauen. Ein Stern markiert neue Matches, deren Details noch nicht angeschaut wurden.

5.5 Einstellungen

Die Einstellungen geben dem Benutzer etwas Kontrolle über die Suche, er kann festlegen, ob sein Mobilgerät automatisch periodisch eine Suche einleiten soll (aktive Suche) und wenn ja, wie oft dies geschehen soll. So kann der Benutzer den Kompromiss zwischen häufiger Suche und Energiesparen selber wählen. Sucht das Gerät nicht regelmässig selber (passive Suche), so beantwortet es trotzdem Anfragen von anderen Geräten und mit "Suche JETZT!" kann eine Suche manuell ausgelöst werden.
Es besteht auch die Möglichkeit unsichtbar zu bleiben: Andere können bei ihrer Suche das Gerät nicht finden, es kann aber vom Gerät aus eine Suche gestartet werden. Dies ist eine Funktion von Bluetooth und kann auf dem Gerät unter "Sichtbarkeit" eingestellt werden.

5.6 Statistiken

Damit der Benutzer eine Ahnung hat, was das BlueDating-Programm so macht, kann er sich einige statistische Werte anzeigen lassen.

5.6.1 Lauftext im Hauptmenü

Im Hauptmenü wird unter dem Fenstertitel ein Lauftext angezeigt, welcher über die Anzahl der Geräte in Reichweite mit BlueDating informiert. Es handelt sich dabei um die Anzahl gesehender BlueDating-Geräte beim letzten Suchvorgang.
Während eine Suche im Gange ist, wird dies im Lauftext mit dem Text "searching..." signalisiert.

5.6.2 Statistik-Anzeige

Etwas mehr Details enthält die Statistikanzeige. Hier lassen sich folgende Werte ansehen:

- Die Anzahl der Suchläufe, die von dem Gerät ausgingen.
- Die Anzahl der Verbindungen zu anderen BlueDating-Programmen.
- Die Anzahl der Matches.

Die Werte lassen sich auf 0 zurücksetzen.

5.7 GUI

Die Abbildungen 5.3 und 5.4 zeigen eine Übersicht über das grafische Oberfläche von BlueDating.

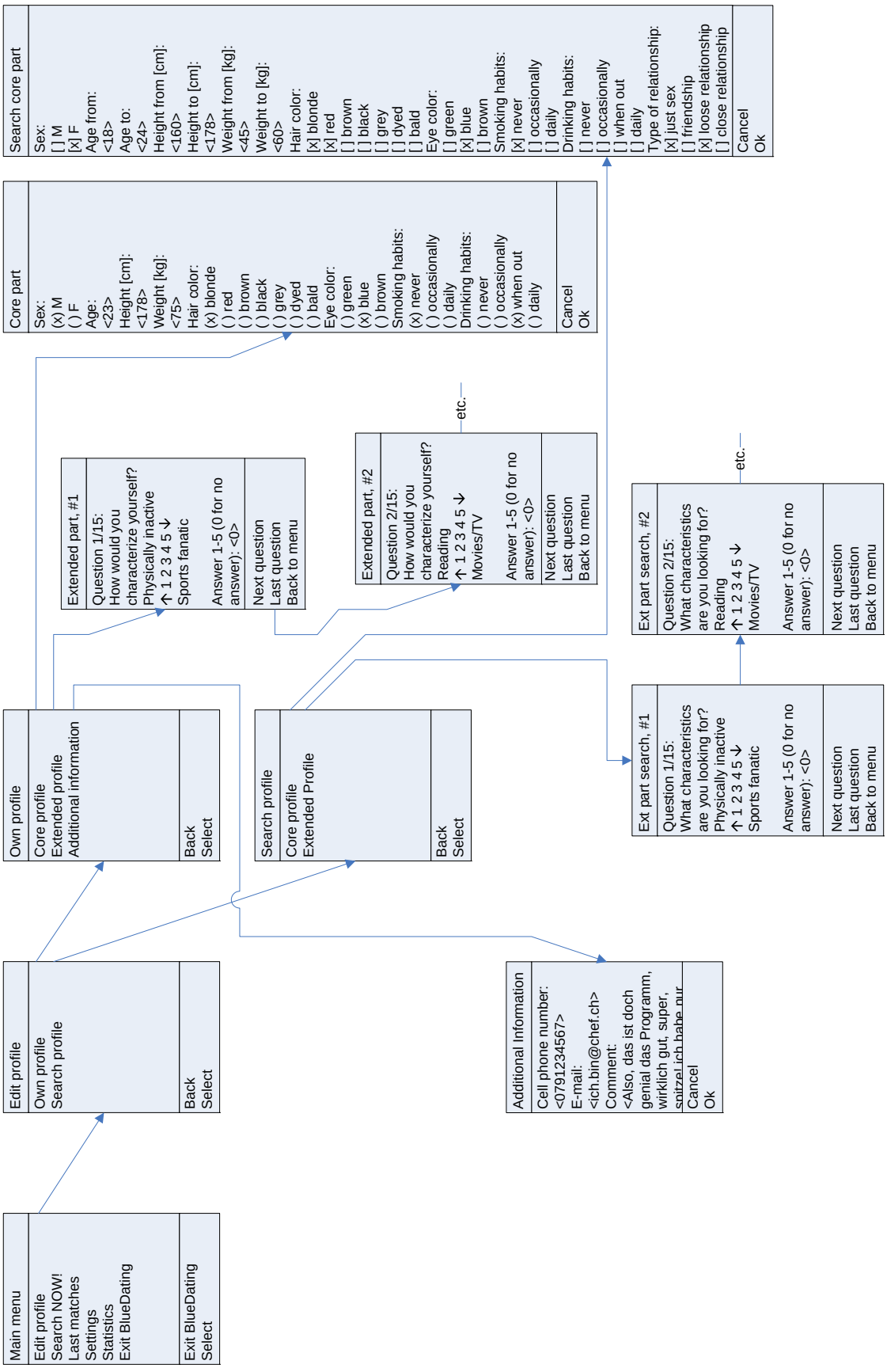


Abbildung 5.3: *Das GUI: Editieren des Profils*

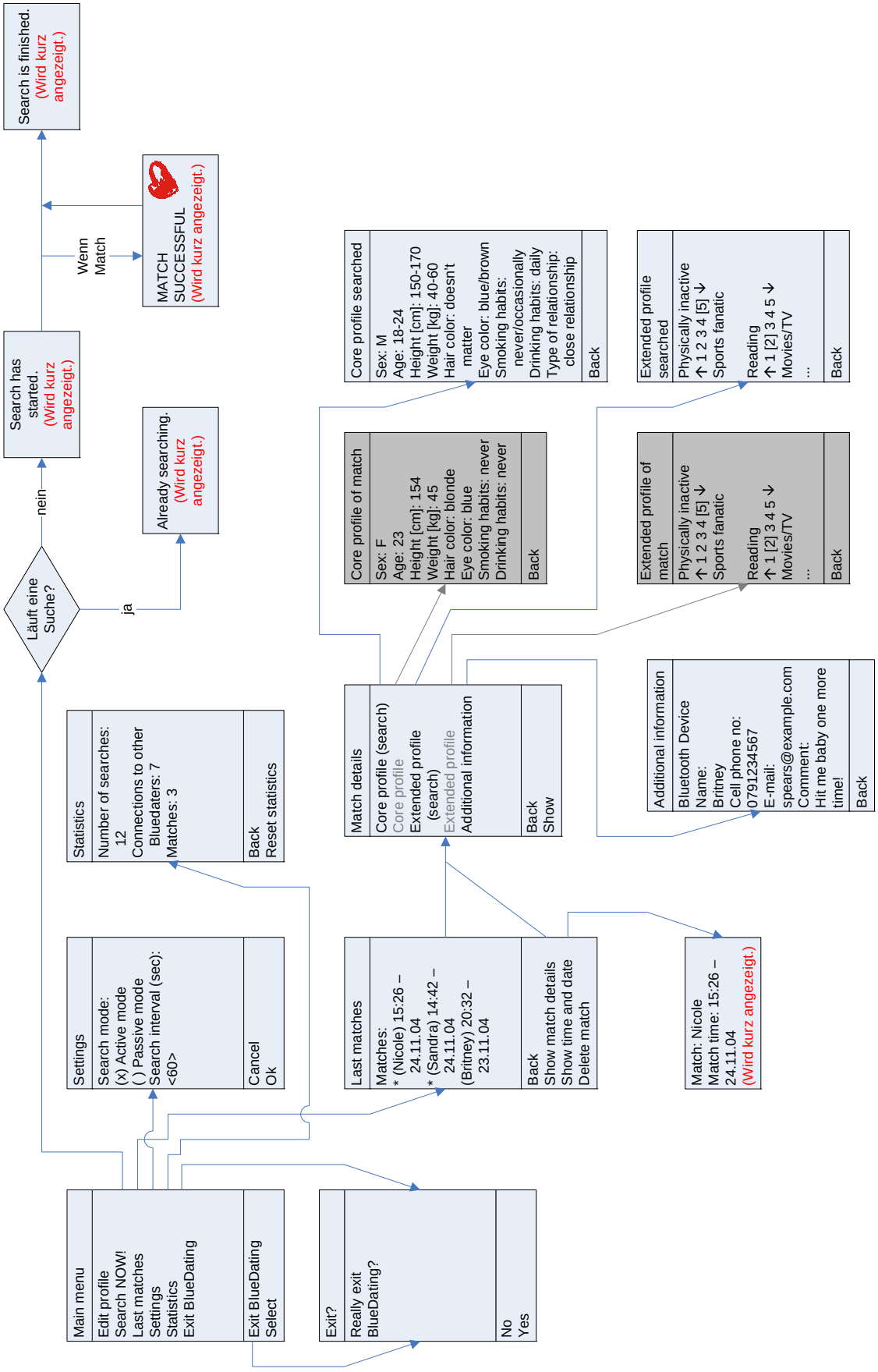


Abbildung 5.4: *Das GUI: Restliche Funktionen*

Kapitel 6

Implementation

6.1 Profile

Neben den Angaben des Benutzers enthält das Profil noch weitere Daten, die wichtigsten darunter sind:

- Die Bluetooth-Adresse des Geräts, zu dem der Match gehört. Über die Bluetoothadresse kann ein Gerät eindeutig identifiziert werden.
- Der Bluetooth-Name des Geräts, zu dem der Match gehört. Dieser Name lässt sich in den Bluetooth-Einstellungen des Geräts wählen.
- Bei den Grundangaben und den erweiterten Angaben jeweils für die eigenen und die Suchangaben ein Flag, ob der Teil schon ausgefüllt bzw. empfangen wurde.

6.1.1 Zusätzliche Angaben mit Fotografie

Natürlich wäre es nützlich, in den zusätzlichen Informationen auch die Möglichkeit zu bieten, eine Fotografie von sich einzufügen. So wüsste das Gegenüber bei einem Match gleich, nach wem er oder sie zu suchen hat. Da aber das zum Filesystem-Zugriff nötige API nicht verbreitet ist, wurde auf dieses Feature vorläufig verzichtet. Im Protokoll ist jedoch bereits, für eine allfällige spätere Implementierung, die Übertragung von Binärdateien wie beispielsweise Bilder, vorgesehen (Siehe Kapitel 6.3.2.6).

6.1.2 Matching

Wenn sich zwei BlueDater begegnen, müssen die beiden Profile mit den im Abschnitt 5.2.1 beschriebenen Daten übertragen und verglichen werden. Dieser Vorgang ist so aufgebaut, dass immer nur die Suchdaten gesendet werden. So werden die persönlichen Daten nicht preisgegeben und die Privatsphäre bleibt geschützt. Die Suchdaten werden dann auf dem anderen Gerät mit dem vorhandenen Profil verglichen. Erfüllt dieses die gesuchten Kriterien, so antwortet das andere Gerät seinerseits mit den Suchdaten seines Benutzers, welche dann wiederum im ersten Gerät mit dem vorhandenen Profil verglichen werden.
Sind die Basisangaben und die erweiterten Angaben erfolgreich gematcht, werden anschliessend die zusätzlichen Informationen in beide Richtungen übertragen.

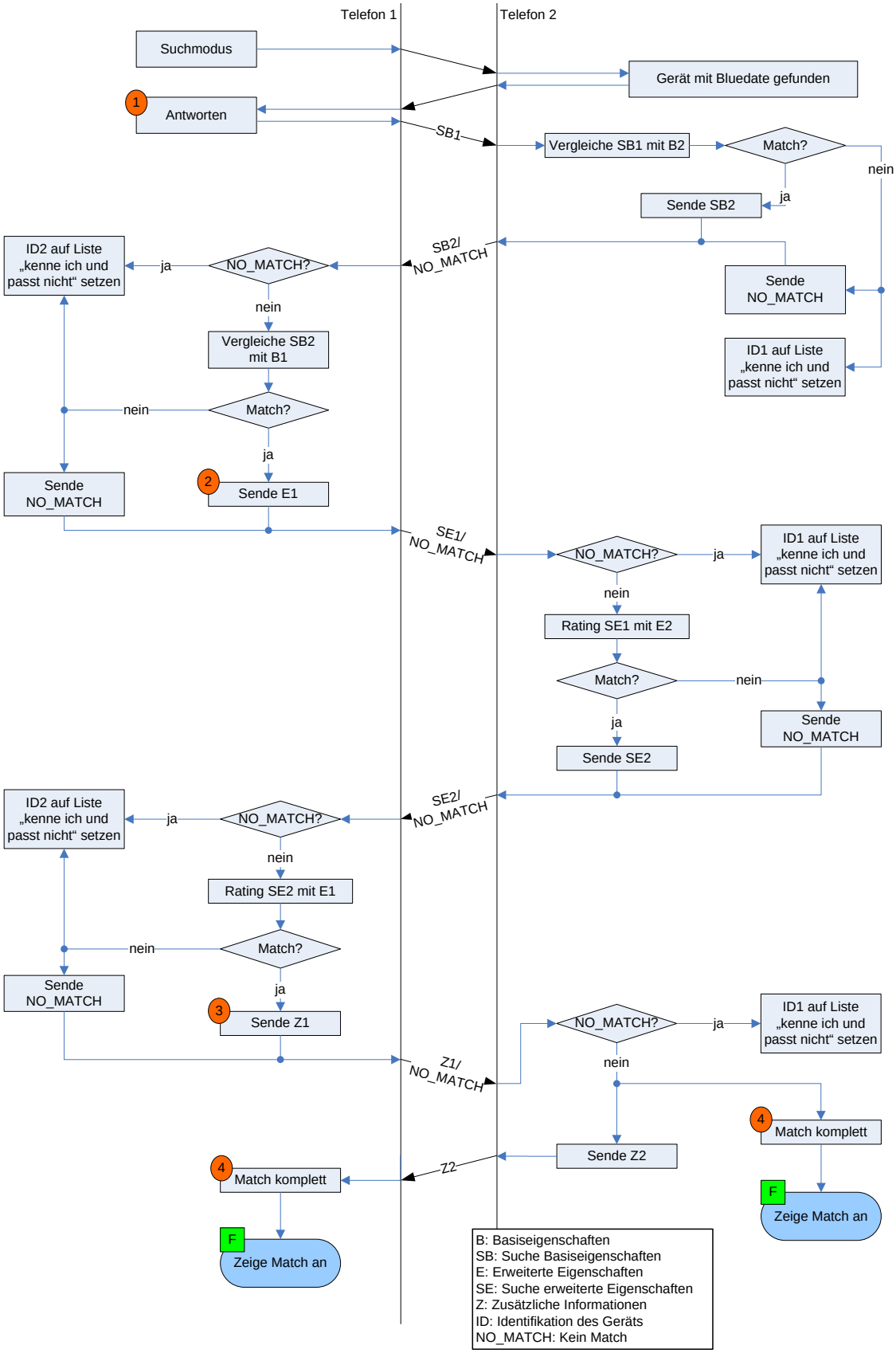


Abbildung 6.1: *Detaillierter Ablauf des Matchings. Die nummerierten roten Kreise zeigen auf, wo dass zwecks Verrechnung unterbrochen werden kann.*

Kommt ein Match nicht zustande, so antwortet das Gerät, anstatt mit dem nächsten Teil der Suchanfrage, mit der Meldung "`NO_MATCH`" und das Matching wird anschliessend abgebrochen.

6.1.2.1 Matching der Basisangaben

Die numerischen Werte werden geprüft, ob sie innerhalb der gesuchten Schranken liegen, wobei berücksichtigt wird, dass auch nur eine obere oder untere Schranke gegeben sein kann.
Die bitweise codierten Werte werden durch bitweises `AND` verglichen, existiert eine Übereinstimmung, so resultiert ein Wert grösser 0. Hat der Benutzer in der Suche für eine Eigenschaft keine Auswahl getroffen (was bedeutet, dass es ihm egal ist), so sind für die betreffende Eigenschaft die Bedingungen für den Match automatisch erfüllt.

6.1.2.2 Matching der erweiterten Angaben

Bei den erweiterten Angaben können sowohl das eigene Profil wie auch die Suchanfrage Lücken aufweisen. Die Implementation ist der Abbildung 6.2 zu entnehmen.

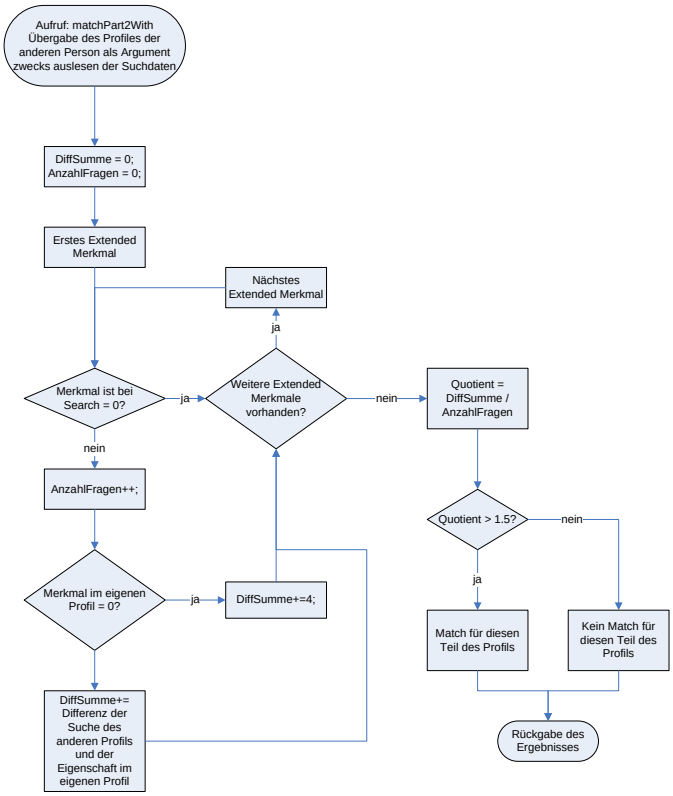


Abbildung 6.2: *Detaillierter Ablauf des Matchings für die erweiterten Eigenschaften.*

6.1.2.2.1 Einstellbare Konstanten

Die Berechnung des Quotienten erfolgt jedoch aufgrund der fehlenden Fliesskomma-Fähigkeit von J2ME ganzzahlig, wobei die nötige Genauigkeit erreicht wird indem der Dividend zuvor mit 100 multipliziert wird. Entsprechend muss als Schwelle für

den Match, die durch die Konstante "`ANSWER_DIFFERENCE_RATIO_LIMIT`" der Profil-Klasse gegeben ist, das 100-fache der durchschnittlichen Abweichung pro Frage angegeben werden. Die Voreinstellung ist 150 (also durchschnittlich 1.5 Punkte Abweichung pro Frage).
Ist eine Frage in der Suchanfrage enthalten, jedoch im zu vergleichenden Profil nicht beantwortet, wird eine vordefinierte Differenz für diese Frage gerechnet. Wie gross diese Straf-Differenz sein soll definiert die Konstante "`DIFFERENCE_IF_SEARCHED_FOR_NOT_GIVEN`". Die Voreinstellung ist 4.

6.1.2.3 Kommerzialisierung

Das Matching läuft in der von uns implementierten Version vollständig zwischen zwei Geräten ab. Soll eine Verrechnung ermöglicht werden, so ist beispielsweise eine Verbindung zu einem Server notwendig. Zu welchen Zeitpunkten ein solcher Unterbruch denkbar ist, zeigen die nummerierten roten Kreise in Abbildung 6.1:

1. Wenn sich zwei Geräte gefunden haben, wird dies an den Server gemeldet. Der Rest des Matchings fände dann auf dem zentralen Server statt, das Bluetooth wird verwendet um festzustellen, dass sich zwei Geräte in nächster Nähe zueinander befinden.

2. Wie Punkt 1, jedoch führen die Mobilgeräte bereits ein Matching für die Basiseigenschaften durch, der Rest geschieht über den Server. Diese Variante wäre vor allem nützlich, wenn die erweiterten Eigenschaften sehr umfangreich ausgelegt würden, wo dann eine Eingabemöglichkeit beispielsweise über ein Webportal eine Erleichterung darstellen würde.

3. Die Geräte finden miteinander heraus, ob ein Match zustande kommt. Ist dies der Fall, werden jedoch keine zusätzlichen Informationen übertragen, stattdessen wird dem Server mitgeteilt, dass ein Match eingetreten ist und gegen Bezahlung der Gebühr lassen sich die erweiterten Angaben des Matches herunterladen und ansehen.

4. Das ganze Matching wird durchgeführt wie in der von uns implementierten Version, auch werden die zusätzlichen Informationen übertragen und es wird angezeigt, dass ein Match zustande gekommen ist. Um sich jedoch die Informationen über den Match anzeigen zu lassen, ist eine kostenpflichtige Freischaltung erforderlich.

Im einfachsten Fall könnte die Abrechnung auch via SMS geschehen: Will sich der Benutzer die Details eines Matches anzeigen lassen, so schickt BlueDating ein SMS mit dem entsprechenden Preis an eine Nummer und zeigt den Match an. Das SMS müsste gar nicht weiter verarbeitet werden, die Gebühren des SMS bilden die Bezahlung.

6.2 GUI

Die grafische Benutzeroberfläche (GUI) von BlueDating ist sehr umfangreich. Wie auch aus den Abbildungen 5.3 und 5.4 hervorgeht, sind vor allem der Editor für das eigene Profil sowie die Liste der Last Matches mit den Match-Details, die sich anzeigen lassen, sehr aufwändig.

Für die Menübildschirme müssen einerseits die anzuzeigenden Elemente und andererseits die Menübefehle vorgängig deklariert und zu den Menüs hinzugefügt werden. Die Menüs sind normalerweise *List* Objekte, die Formulare *Form* Objekte und die Alarmfensterchen *Alert* Objekte. Wird über die Tastatur des Mobilgeräts ein Befehl ausgewählt, so wird automatisch die Funktion *CommandListener* ausgeführt, welcher die Reaktion auf den Menübefehl auslöst.
Im Anschluss werden einige ausgewählte Details des GUIs erläutert.

6.2.1 Alarme

Das GUI stellt eine Methode zur Verfügung, mit der dem Benutzer ein Fenster mit einer frei wählbaren Meldung angezeigt werden kann. So besteht die Möglichkeit, den Benutzer über Ereignisse oder Fehler zu informieren. Diese Methode sorgt auch dafür, dass diese Meldung mindestens für eine gewisse Zeitdauer angezeigt wird, auch wenn das Programm gleich anschliessend ein anders Menü laden will, welches dann die Alarmmeldung überschreiben würde. Die Dauer lässt sich in der *Settings* Klasse mit der Konstante `ALERT_WAITTIME_MILLIS` festlegen.
Leider ist es uns nicht gelungen zu garantieren, dass ein Alarm wirklich angezeigt wird. In gewissen Situationen werden diese aus unerfindlichen Gründen schlichtweg nicht angezeigt. Ob das Alarmfenster auf dem Bildschirm erschien, hing in unseren Versuchen von der Stelle im Code ab und schien von nachfolgenden Instruktionen beeinflusst zu werden. Mit der Absicht, die Ursache für diese Unterschlagung zu finden, führten wir die betreffende Stelle mit dem Debugger daher zeilenweise aus, doch der Alarm fand seinen Weg auf den Bildschirm trotzdem nie.

6.2.2 Profileditor-Menü

Wie bereits erwähnt ist dieses Menü sehr umfangreich. Während die Menüs der Basisangaben und der zusätzlichen Informationen im Voraus definiert werden, werden die Eingabemasken der erweiterten Angaben für jede Frage neu aufgebaut, wobei die Worte der Begriffspaare aus je einem Array geladen werden.

6.2.3 Letzte Matches

Die Anzeige der letzten Matches wird jeweils neu generiert, wenn ein Match hinzukommt oder entfernt wird. Bei den Match-Details werden nur die vorhandenen Profilteile angezeigt, die eigenen Daten eines fremden Profils werden ja aus Datenschutzgründen nicht übertragen und fehlen in der Liste, wären sie jedoch vorhanden, stünden sie jedoch automatisch im Menü zur Auswahl.

6.3 Protokoll Spezifikation

Die Abbildung 6.3 zeigt die verschiedenen Datenblöcke des Protokolls.

6.3.1 Der Protokoll-Kopf

Der Protokoll-Kopf ist 2 Bytes lang. Er beinhaltet die Versionsnummer des Protokolls, die Message Type ID und die Anzahl der folgenden Datenfelder.

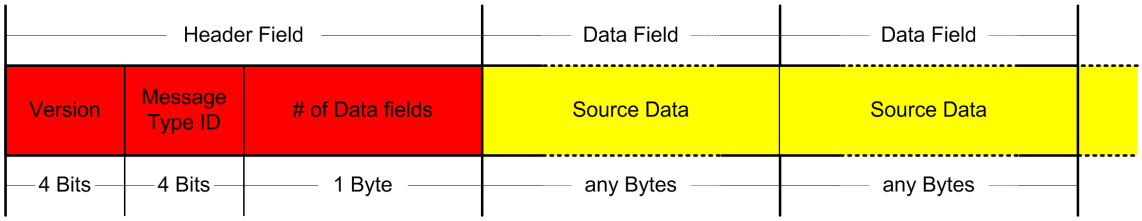


Abbildung 6.3: *Protokoll Spezifikation*

6.3.1.1 Die Versionsnummer

Diese 4 Bits repräsentieren die Versionsnummer des Protokolls, welche von der Applikation gebraucht wird, um zu testen, ob ein aktuelles oder ein veraltetes Protokoll vorliegt.

6.3.1.2 Die Message Type ID

Die Tabelle 6.1 beschreibt die verschiedenen Message Type IDs.

Message Type ID [hex]	Bezeichnung	Beschreibung
0x0	SEARCH_CORE	Sende gesuchte Basiseigenschaften
0x1	SEARCH_EXT	Sende gesuchte erweiterte Eigenschaften
0x2	OWN_CORE	Sende eigene Basiseigenschaften
0x3	OWN_EXT	Sende eigene erweiterte Eigenschaften
0x4	ADD_INFO	Sende zusätzliche Informationen
0x5	FILE_XFER	File transfer
0xC	REQ	Anforderung von Daten
0xD	TEST	Übertragungstest
0xE	NO_MATCH	Es fand kein Match statt
0xF	ERROR	Empfänger hatte einen Fehler

Tabelle 6.1: *Message Type ID*

6.3.1.3 Die Anzahl der Datensätze

Diese 8 Bits geben an wie viele Datenfelder folgen. Es sind demnach maximal 255 folgende Datenfelder möglich.

6.3.2 Das Datenfeld

6.3.2.1 Das Datenfeld von "Sende gesuchte Core Characteristics"

Die Abbildung 6.4 zeigt das Datenfeld von "Sende gesuchte Core Characteristics".

6.3.2.1.1 Die Attribute Type ID

Die Tabelle 6.2 beschreibt die verschiedenen Attribute Type IDs von "Sende gesuchte Core Characteristics".

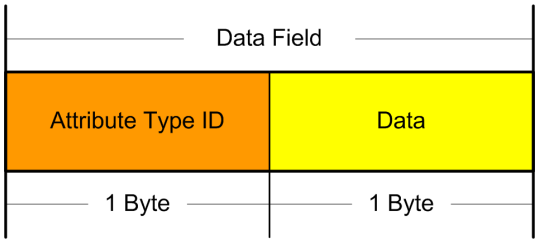


Abbildung 6.4: *Das Datenfeld von "Sende gesuchte Core Characteristics"*

Attribute Type ID [hex]	Bezeichnung	Beschreibung
0x00	SEX	Geschlecht
0x01	AGE_FROM	Alter ab
0x02	AGE_TO	Alter bis
0x03	HEIGHT_FROM	Grösse ab
0x04	HEIGHT_TO	Grösse bis
0x05	WEIGHT_FROM	Gewicht ab
0x06	WEIGHT_TO	Gewicht bis
0x07	HAIR	Haarfarbe
0x08	EYE	Augenfarbe
0x09	SMOKE	Rauchgewohnheiten
0x0A	DRINK	Trinkgewohnheiten
0x0B	RELATIONSHIP	Beziehungsart

Tabelle 6.2: *Attribute Type IDs von "Sende gesuchte Core Characteristics"*

6.3.2.1.2 Die Daten

Das Datenbyte kann auf zwei verschiedene Arten aufgebaut sein, was abhängig von der Attribute Type ID ist. Entweder bitweise oder das ganze Byte als einen unsigned Interger.

Bitweise

Dies wird bei Eigenschaften gebraucht, welche eine Auswahl an verschiedenen Kriterien zulassen. Jedem Kriterium wird ein Bit zugeordnet, welches auf 1 gesetzt werden kann, wenn es zutrifft. Falls der Benutzer keine Eigenschaft auswählt, steht das Byte auf `0xFF`, was als "Egal" interpretiert wird. Folgende Attribute Type IDs verwenden die bitweise Darstellung:

Geschlecht
Die Abbildung 6.5 zeigt die Daten von "Geschlecht".

Haarfarbe
Die Abbildung 6.6 zeigt die Daten von "Geschlecht".

Augenfarbe
Die Abbildung 6.7 zeigt die Daten von "Augenfarbe".

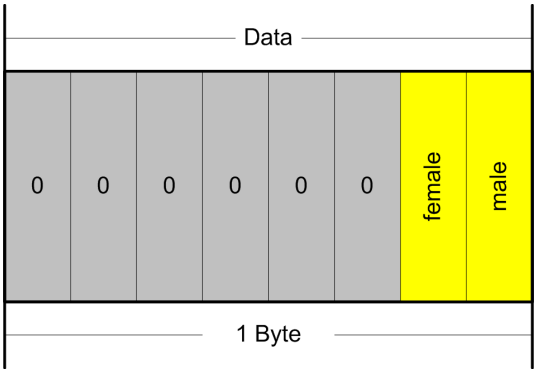


Abbildung 6.5: *Die Daten von "Geschlecht"*

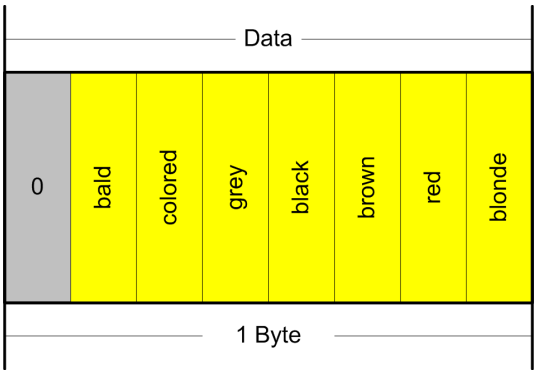


Abbildung 6.6: *Die Daten von "Haarfarbe"*

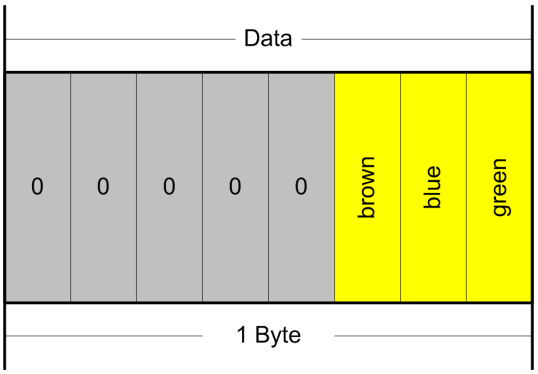


Abbildung 6.7: *Die Daten von "Augenfarbe"*

Rauchgewohnheiten
Die Abbildung 6.8 zeigt die Daten von "Rauchgewohnheiten".

Trinkgewohnheiten
Die Abbildung 6.9 zeigt die Daten von "Trinkgewohnheiten".

Beziehungsart
Die Abbildung 6.10 zeigt die Daten von "Beziehungsart".

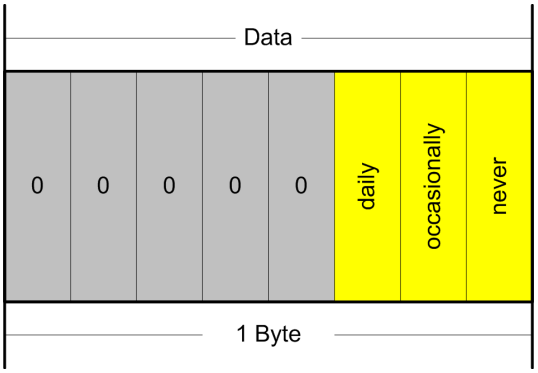


Abbildung 6.8: *Die Daten von "Rauchgewohnheiten"*

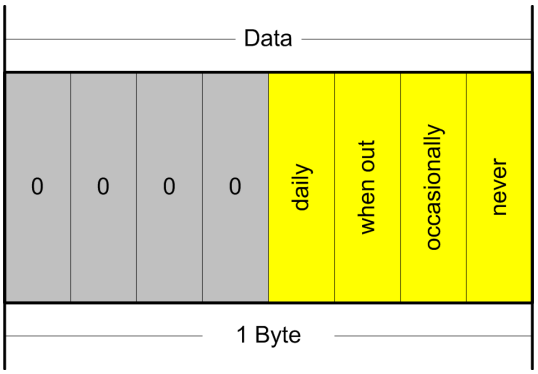


Abbildung 6.9: *Die Daten von "Trinkgewohnheiten"*

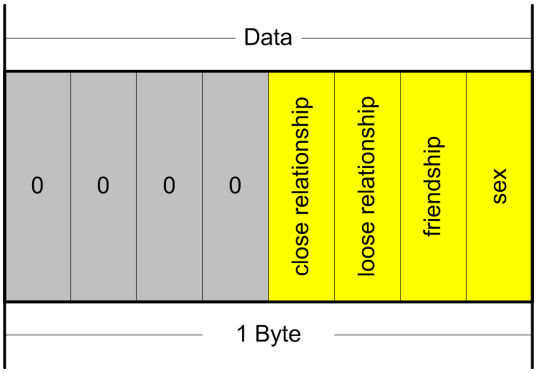


Abbildung 6.10: *Die Daten von "Beziehungsart"*

Byteweise

Das Datenbyte wird als ganzes Byte gesetzt, das heisst es wird einfach der Wert als `8 Bit unsigned Integer` geschrieben. Falls der Benutzer die Auswahl "Egal" trifft, wird das Byte auf `0x00` gesetzt. Folgende Attribute Type IDs verwenden die byteweise Darstellung:

- Age from/to
- Height from/to
- Weight from/to

Es sind Werte im Bereich `0x01` bis `0xFE` (254) erlaubt, da der Wert `0x00` für die "Egal"-Auswahl reserviert ist.

6.3.2.2 Das Datenfeld von "Sende gesuchte Extended Characteristics"

Das Datenfeld von "Sende gesuchte Extended Characteristics" hat die gleiche Struktur wie das Datenfeld von "Sende gesuchte Core Characteristics" in Abbildung 6.4.

6.3.2.2.1 Die Attribute Type ID

Die Tabelle 6.3 beschreibt die verschiedenen Attribute Type IDs von "Sende gesuchte Extended Characteristics".

Attribute Type ID [hex]	Bezeichnung	Beschreibung
0x00	SPORTS	Bewegungsfaul / Sportfanatiker
0x01	BOOKS_TV	Lesen / Kino, TV
0x02	PARTY_RELAX	Party, Ausgang / Ruhiger Abend
0x03	TRAMP_HOTEL	Trampen / 5-Sterne-Hotel
0x04	NIGHT_DAY	Nachtmensch / Früh ins Bett
0x05	MORNING	Frühaufsteher / Morgenmuffel
0x06	HARMONY	Harmonie / Energie
0x07	ARMANI	Armani / 2nd Hand
0x08	PEN_COMP	Papier und Bleistift / Computerfreak
0x09	FAST_REST	Take Out, Fast Food / Restaurant
0x0A	WORLD_LOCAL	Weltentdecker / Stubenhocker
0x0B	CITY	Stadtmensch / Landmensch
0x0C	ZOO	Kein Haustier / "Zoo" daheim
0x0D	IN_OUTDOOR	Indoor / Outdoor
0x0E	GROUP	Einzelgänger / Gruppenmensch

Tabelle 6.3: *Attribute Type IDs von "Sende gesuchte Extended Characteristics"*

6.3.2.2.2 Die Daten

Alle Daten werden byteweise geschrieben, da ja die Auswahl des Benutzers ein Wert zwischen 0 und 5 ist. 0 steht dabei für "Egal".

6.3.2.3 Das Datenfeld von "Sende eigene Core Characteristics"

Das Datenfeld von "Sende eigene Core Characteristics" hat die gleiche Struktur wie das Datenfeld von "Sende gesuchte Core Characteristics" in Abbildung 6.4.

6.3.2.3.1 Die Attribute Type ID

Die Tabelle 6.4 beschreibt die verschiedenen Attribute Type IDs von "Sende eigene Core Characteristics".

Attribute Type ID [hex]	Bezeichnung	Beschreibung
0x00	SEX	Geschlecht
0x01	AGE	Alter
0x03	HEIGHT	Grösse
0x05	WEIGHT	Gewicht
0x07	HAIR	Haarfarbe
0x08	EYE	Augenfarbe
0x09	SMOKE	Rauchgewohnheiten
0x0A	DRINK	Trinkgewohnheiten

Tabelle 6.4: *Attribute Type IDs von "Sende eigene Core Characteristics"*

6.3.2.3.2 Die Daten

Das Datenbyte kann auf zwei verschiedene Arten aufgebaut sein, was abhängig von der Attribute Type ID ist. Entweder bitweise oder das ganze Byte als einen unsigned Interger.

Bitweise

Dies wird bei gebraucht, welche eine Auswahl an verschiedenen Kriterien zulassen. Jedem Kriterium wird ein Bit zugeordnet, welches auf 1 gesetzt werden kann, wenn es zutrifft. Die Daten haben die gleiche Struktur wie im Abschnitt 6.3.2.1.2, jedoch ohne die Möglichkeit den Wert für "Egal" zu wählen.

Byteweise

Das Datenbyte wird als ganzes Byte gesetzt, das heisst es wird einfach der Wert als `8 Bit unsigned Integer` geschrieben. Folgende Attribute Type IDs verwenden die byteweise Darstellung:

- Age
- Height
- Weight

Es sind Werte im Bereich `0x0` bis `0xFE` (254) erlaubt.

6.3.2.4 Das Datenfeld von "Sende eigene Extended Characteristics"

Das Datenfeld von "Sende eigene Extended Characteristics" sieht gleich aus wie das Datenfeld von "Sende gesuchte Extended Characteristics" im Abschnitt 6.3.2.2, mit der Ausnahme, dass 0 nicht für "Egal" sondern für "Keine Angabe" steht.

6.3.2.5 Das Datenfeld von "Sende Additional Information"

Die Abbildung 6.11 zeigt das Datenfeld von "Sende Additional Information".

6.3.2.5.1 Die Attribute Type ID

Die Tabelle 6.5 beschreibt die verschiedenen Attribute Type IDs von "Sende Additional Information".

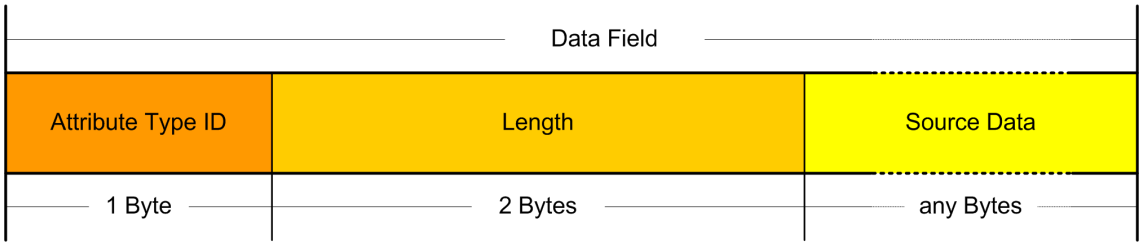


Abbildung 6.11: *Das Datenfeld von "Sende Additional Information"*

Attribute Type ID [hex]	Bezeichnung	Beschreibung
0x00	HANDY	Nummer des Mobiltelefons
0x01	EMAIL	E-mail Adresse
0x02	COMMENT	Kommentar

Tabelle 6.5: *Attribute Type IDs von "Sende Additional Information"*

6.3.2.5.2 Die Länge

Da bei den Einträgen der "Additional Information" die Angaben in der Länge variieren, enthalten diese 2 Bytes die Länge des nachfolgenden Datenblocks. Mit 2 Bytes sind Datenblocklängen bis zu 65 kB möglich.

6.3.2.5.3 Die Daten

Die Werte werden Byteweise geschrieben. Ein Byte entspricht einem Zeichen (CHAR).

6.3.2.6 Das Datenfeld von "File Transfer"

Die Abbildung 6.12 zeigt das Datenfeld von "File Transfer". Es unterscheidet sich von "Sende Additional Information" im vorigen Abschnitt dadurch, dass für die Längenangabe 4 Byte vorhanden sind, was Engpässen vorbeugt (bis 4.2 GB möglich).

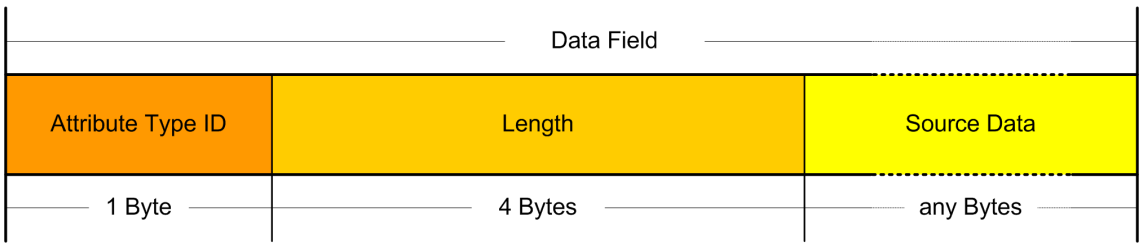


Abbildung 6.12: *Das Datenfeld von "File Transfer"*

Wegen der schlechten Unterstützung (siehe Abschnitt 3.4) des Dateisystemzugriffs aus Java auf heutigen Mobilgeräten ist die Funktion "File Transfer" nicht implementiert. Die Spezifikation der Dateiübertragung im Protokoll soll jedoch eine schnelle spätere Implementation erleichtern.

6.3.2.6.1 Die Attribute Type ID

Die Tabelle 6.6 beschreibt die verschiedenen Attribute Type IDs von "File Transfer".

Attribute Type ID [hex]	Bezeichnung	Beschreibung
0x00	PICTURE	Bild des Benutzers

Tabelle 6.6: *Attribute Type IDs von "File Transfer"*

6.3.2.7 Das Datenfeld von "Anforderung von Daten"

Das Datenfeld für die "Anforderung von Daten" (Abbildung 6.13) ist zwei Byte gross, jedoch wird das zweite Byte ignoriert, je nachdem welche Message Type ID die angeforderten Daten haben. Wo dies zutrifft, ist aus Tabelle 6.7 ersichtlich. Jene Message Type IDs, welche in der Tabelle nicht aufgeführt sind (jene, deren Nummern von 0xF absteigen), können nicht abgerufen werden.

Message Type ID	Berücksichtigte Länge des Datenfeldes
SEARCH_CORE	1
SEARCH_EXT	1
OWN_CORE	1
OWN_EXT	1
ADD_INFO	2
FILE_XFER	2

Tabelle 6.7: *Anzahl berücksichtigter Bytes des Datenfeldes von "Anforderung von Daten" in Abhändigkeit der Message Type ID*

Das erste Byte des Datenfeldes entspricht jeweils der Message Type ID der angeforderten Informationen (inklusive der Protokollversion, da dies die Eindeutigkeit der Message Type ID garantiert). Bei jenen Message Type IDs, bei denen das Datenfeld nur 1 Byte lang ist, gilt ein Abruf also jeweils für alle Informationen, die unter die entsprechende Message Type ID fallen.
Das zweite Byte des Datenfeldes, wenn es denn eine Bedeutung hat, steht für die Attribute Type ID der angeforderten Informationen. Bei gewissen Message Type IDs muss es jedoch ignoriert werden.

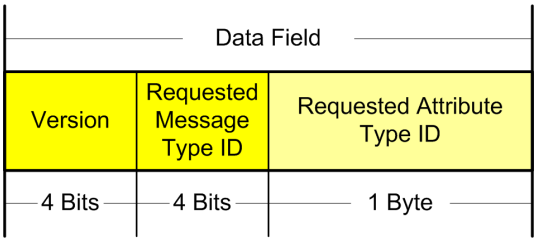


Abbildung 6.13: *Das Datenfeld von "Anforderung von Daten"*

Die Möglichkeit, Daten von einem anderen Gerät abzurufen, ist in BlueDating nicht implementiert, da unser Konzept davon ausgeht, dass der eigene Teil des Profils (mal abgesehen von den zusätzlichen Angaben) nie übertragen werden soll. Die Spezifikation der "Anforderung von Daten" im Protokoll soll dabei helfen, dass ein Abruf von Daten schnell eingebaut werden könnte.

6.3.2.8 Das Datenfeld von "Transmission Test"

Das Datenfeld von "Transmission Test" ist ein Byte gross und enthält einen wählbaren Wert um die Verbindung zu testen.

6.3.2.9 Das Datenfeld von "Fail"

Das Datenfeld für die Message Type ID "Fail" ist leer. Das Protokoll für eine "Fail"-Antwort besteht nur aus dem Protokoll Kopf, wobei die Anzahl der folgenden Datensätze Null ist.

6.3.2.10 Das Datenfeld von "Error"

Das Datenfeld für die Message Type ID "Error" besteht aus einem Byte, welches die Error Nummer enthaltet.
Die Tabelle 6.8 beschreibt die verschiedenen Error Nummern.

6.4 Last Matches

Um all die erfolgten Matches zu verwalten, wurde der *Last Matches* Vektor definiert. Dieser enthält Objekte vom Typ Match. Der Typ Match ist als Klasse definiert und enthält folgende Attribute:

- `String DeviceBluetoothAddress`
- `Calendar calendar`
- `boolean isNew`
- `String friendlyName`

Die `DeviceBluetoothAdresse` ist die Referenz zum anderen Bluetooth Gerät, mit dem der Match stattfand. `calendar` beinhaltet der Zeitpunkt des Matches. `isNew` ist *true*, falls bei diesem Match die Details noch nie betrachtet wurden. Der `friendlyName`[1] bezieht sich ebenfalls auf das andere Gerät, mit dem der Match erfolgte.
Der Zugriff auf diesen Vektor ist nur über vordefinierte Methoden möglich, welche das lesen, schreiben und löschen einzelner Matches gestatten.

[1] Dies ist ein Name den man jedem Bluetooth-Gerät zuteilen kann, und dient der besseren Verständlichkeit gegenüber der Bluetoothaddresse als simple lange Zahl

Attribute Type ID [hex]	Bezeichnung	Beschreibung
0x00	eNO_ERROR	Kein Fehler
0x01	eREAD_ERROR	Lese Fehler
0x02	eWRITE_ERROR	Schreib Fehler
0x03	eWRONG_PROTOCOL_-VERSION	Der Sender benutzte eine falsche Protokoll Version
0x04	eNOT_RECEIVED_-CORRECT_TEST_BYTE	Der Sender benutzte ein falsches Test Byte
0x05	eNOT_RECEIVED_ALL_-SEARCH_CORE_-CHARACTERISTICS	Habe nicht alle gesuchten Basiseigenschaften erhalten
0x06	eNOT_RECEIVED_ALL_-OWN_CORE_-CHARACTERISTICS	Habe nicht alle eigenen Basiseigenschaften erhalten
0x07	eNOT_RECEIVED_ALL_-SEARCH_EXTENDED_-CHARACTERISTICS	Habe nicht alle gesuchten erweiterten Eigenschaften erhalten
0x08	eNOT_RECEIVED_ALL_-OWN_EXTENDED_-CHARACTERISTICS	Habe nicht alle eigenen erweiterten Eigenschaften erhalten
0x09	eNO_TEST	Habe keinen Test erhalten
0x0A	eNO_SEARCH_CORE_-CHARACTERISTICS_-RECEIVED	Habe die gesuchten Basiseigenschaften nicht erhalten
0x0B	eNO_OWN_CORE_-CHARACTERISTICS_-RECEIVED	Habe die eigenen Basiseigenschaften nicht erhalten
0x0C	eNO_SEARCH_EXTENDED_-CHARACTERISTICS_-RECEIVED	Habe die gesuchten erweiterten Eigenschaften nicht erhalten
0x0D	eNO_OWN_EXTENDED_-CHARACTERISTICS_-RECEIVED	Habe die eigenen erweiterten Eigenschaften nicht erhalten
0x0E	eNO_ADDITIONAL_-INFORMATION_RECEIVED	Habe die zusätzliche Informationen nicht erhalten

Tabelle 6.8: *Error Nummern*

6.5 Known Devices

Um all die bekannten Geräte im Umkreis zu verwalten, wurde der *Known Devices* Vektor definiert. Dieser enthält Objekte vom Typ Device. Der Typ Device ist als Klasse definiert und enthält folgende Attribute:

- `String DeviceBluetoothAddress`
- `Calendar calendar`

Die `DeviceBluetoothAdresse` ist die Referenz zum anderen Bluetooth Gerät. Das Attribut `calendar` beinhaltet der Zeitpunkt des letzten Kontakts.
Der Zweck dieser *Known Devices* ist es, sich die bereits abgefragten Geräte eine bestimmte Zeit lang zu merken. Dies hat den Vorteil, dass wenn man ein Gerät entdeckt, mit diesem keine komplette Kommunikation mehr starten muss, da dies bereits vor kurzer Zeit geschah. Die Zeitdauer der Gültigkeit kann mit der Konstante `DEVICE_VALIDITY_MILLIS` gewählt werden, standardmässig ist sie auf eine Stunde gestellt. Man könnte diesen Wert auch massiv erhöhen um Bluetooth-Übertragungen und somit Akku zu sparen, aber man muss sich im Klaren sein, dass der Gegenüber seine Daten auch ändern kann, also vielleicht plötlich eine anderes Ergebnis hervorrufen könnte. Allzu grosse Werte sind daher nicht sinnvoll. Einträge die älter sind als die angegebene Gültigkeitsdauer werden automatisch aus der Liste entfernt.
Der Zugriff auf diesen Vektor ist ebenfalls nur über vordefinierte Methoden möglich, welche das lesen, schreiben und löschen einzelner Matches gestatten.

6.6 Kommunikationsablauf

6.6.1 Die Client-Server Struktur

Beim Start von BlueDating wird ein BlueDating-Server erstellt, der auf ankommende Anfragen antworten kann. Dieser Server ist von anderen Geräten auffindbar, sofern das eigene Gerät nicht unsichtbar ist oder das Bluetooth gar abgeschaltet wurde. Ist Bluetooth beim Start deaktiviert erfolgt eine Meldung an den Benutzer, da ja dadurch der Server nicht gestartet werden konnte.
Der Client wird gestartet sobald eine Suche beginnt. Es ist also immer das Gerät, das die Suche startet der Client. Die Geräte die ihm antworten sind die Server. Es kann also vorkommen, dass auf einem Gerät zur selben Zeit der Client und der Server aktiv sind. Aus diesem Grund wurden beide in separaten Threads implementiert.
Ein Client kann mehrere Server aufs Mal finden. Anstatt dass er alle diese Server der Reihe nach abarbeitet, wird für jeden gefundenen Server ein *Server Processor* Thread gestartet, der die weitere Kommunikation durchführt. Das selbe gilt für den Server, der von mehreren Clients pro Zeit aufgerufen werden kann. Dieser startet für jeden anfragenden Client einen *Client Processor* Thread, der die Kommunikation weiterführt. Die Abbildung 6.14 veranschaulicht dieses Verhalten.

6.6.2 Der Verbindungsaufbau

Wie schon beschrieben beginnt der Client die Verbindung, indem er nach vorhandenen BlueDating Servern Ausschau haltet. Wie dies geschieht, ist im Abschnitt 4.2.2 beschrieben. Aus der somit erhaltenen *StreamConnection* kann man die *Bluetooth Device Address* auslesen. Diese Adresse wird nun mit den schon vorhandenen

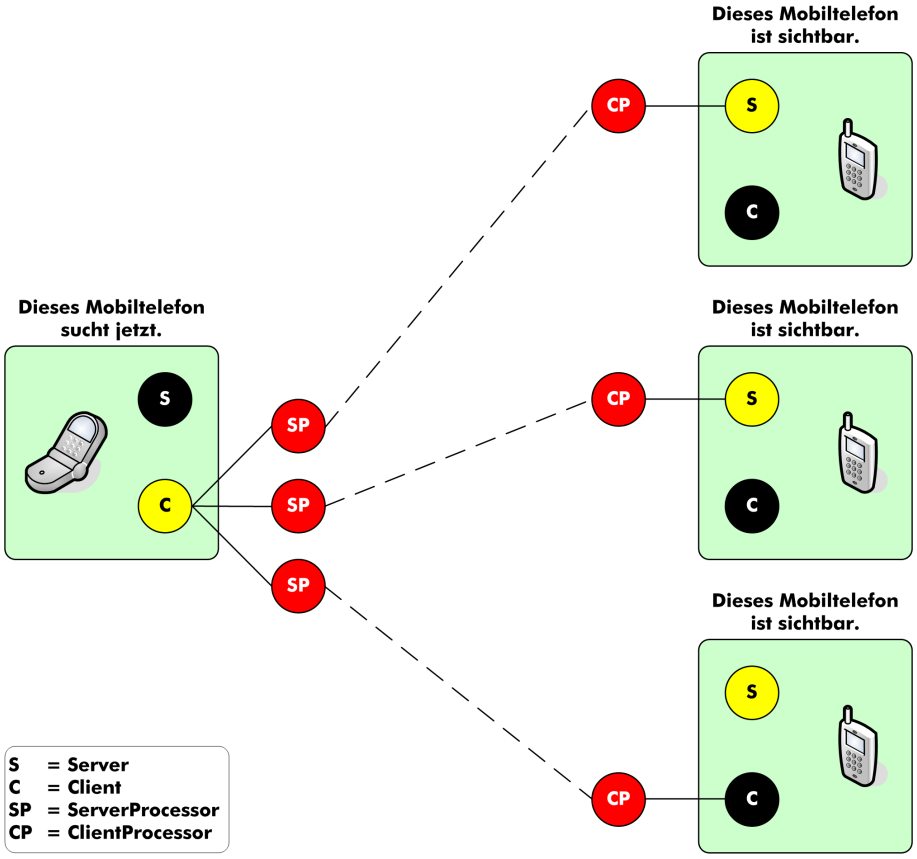


Abbildung 6.14: *Die Threads während der Kommunikation*

Einträgen in den *Known Devices* (siehe Abschnitt 6.5) verglichen. Ist das andere Gerät schon in dieser Liste vorhanden, wird die Kommunikation zu diesem Gerät abgebrochen. Der Server hat zum Zeitpunkt, wo der Client die *StreamConnection* erhielt auch schon einen *Client Processor* Thread gestartet, welcher jetzt auch beendet wird.
Ebenfalls wird eine Überprüfung der *Last Matches* durchgeführt. Ist das andere Gerät bereits in den *Last Matches* vorhanden, wird die Kommunikation ebenfalls abgebrochen. Die suchende Person möchte diesen Match ja auf jeden Fall behalten, sonst hätte Sie ihn gelöscht. Es wird deshalb keine neue Suche durchdeführt.
Ist diese Hürde überwunden, wird der Client für diesen Server einen *Server Processor* Thread starten. Zum jetztigen Zeitpunkt ist der Server und der Client bereit Daten auszutauschen. Die beiden kommunizieren ab jetzt so wie in Abbildung 6.1 beschrieben ist.

6.6.3 Kommunikation im Detail

Um den Kommunikationsablauf im Detail zu erklären, wird er am Beispiel eines Verbindungstests erläutert. Die Abbildung 6.15 zeigt den genauen Ablauf um einen Verbindungstest durchzuführen. Der Client sendet eine Message mit dem Type *TEST* und wartet auf eine Antwort des Servers. Der Server empfängt diese und sendet sie zurück.
Die 5 Methoden, welche zur Verwendung kommen, werden von der Protokoll-Klasse zur Verfügung gestellt.
Das Ablaufdiagramm zeigt alle möglichen Verzweigungen auf, die entstehen könnten

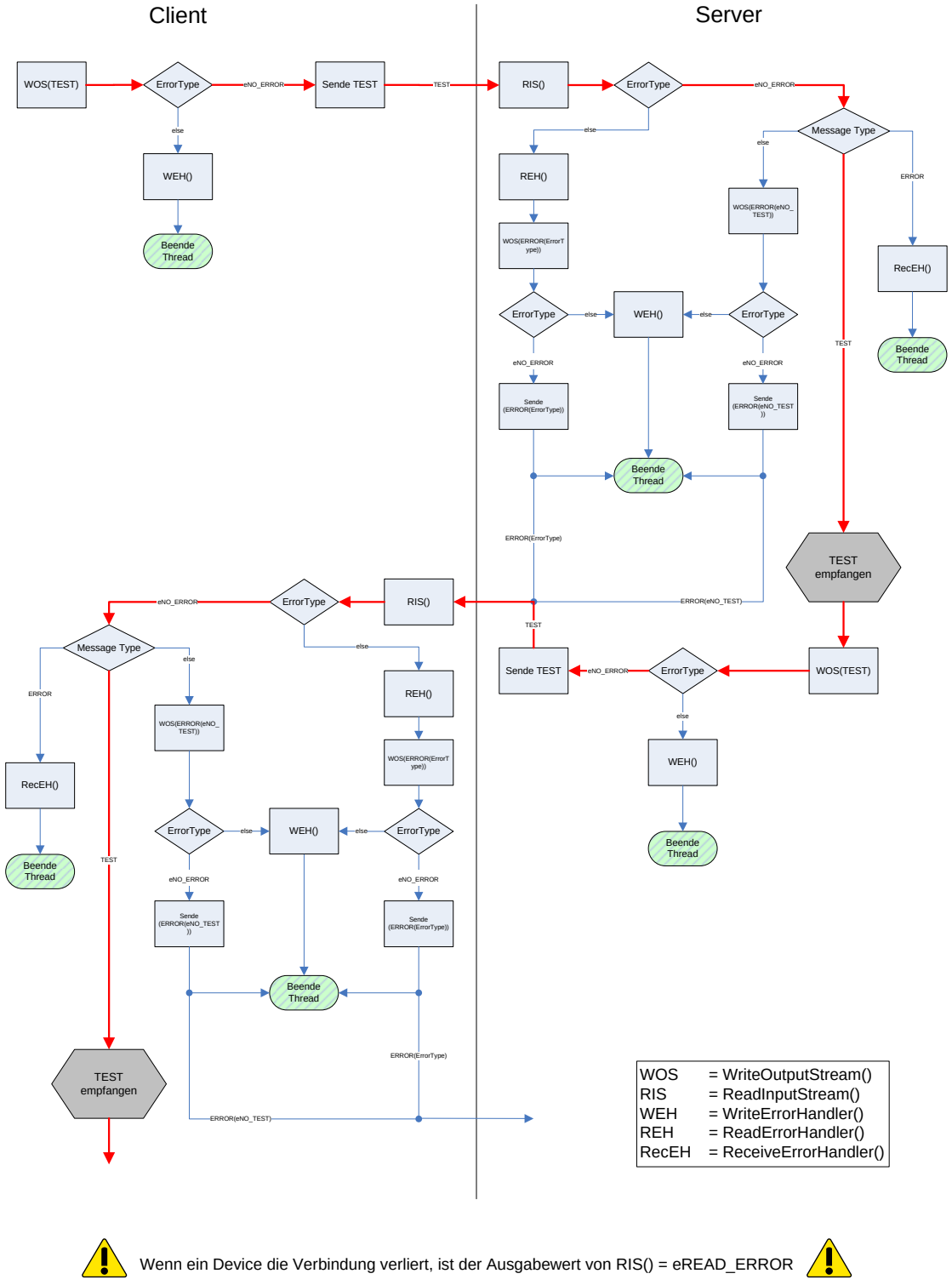


Abbildung 6.15: *Kommunikationsablauf am Beispiel des Message Types TEST*

falls ein Teil falsch übertragen wurde. Es zeigt ebenfalls das Verhalten der Software nachdem ein Fehler entdeckt wurde. Falls kein Fehler auftritt erfolgt die Kommunikation wie nach dem roten Pfad in der Abbildung 6.15.

Bei einem `read()`, welches von der `ReadInputStream()`-Methode aufgerufen wird,

ist der aktuelle Thread so lange blockiert, bis Daten auf diesem Stream anliegen. Verliert jetzt dieser wartende Thread die Verbindung zum Gegenüber (zum Beispiel weil die beiden Geräte nun zu weit auseinander liegen), wäre dieser Thread für immer blockiert. Aus diesem Grund liefert die `read()`-Methode dem Thread bei einem Verbindungunterbruch einen `eREAD_ERROR` Fehler Typ zurück, damit der Thread vollständig beendet werden kann.

6.7 Profil-Abspeicherungs-Spezifikation

6.7.1 Organisation der Profile in RecordStores

MIDP bietet zur persistenten Speicherung von Anwendungsdaten sogenannte *RecordStores* an. Diese benannten Datensammlungen bestehen aus einer Reihe einzelnen Datensätze, den sogenannten *Records*.
Um ein Profil abzuspeichern wird jeweils ein eigener RecordStore verwendet, wobei für jeden Profileintrag ein neuer Record verwendet wird. Abbildung 6.16 veranschaulicht dies.

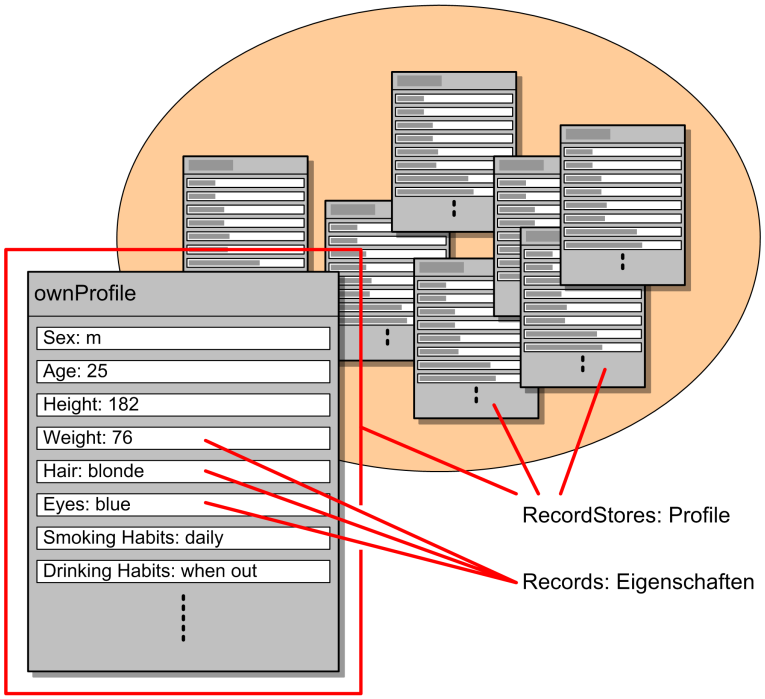


Abbildung 6.16: *Organisation der Profildaten in RecordStores*

6.7.2 RecordStore und Übertragungsprotokoll

An die RecordStores - zur permanenten Abspeicherung von Profilen - und an das Protokoll zur Übertragung von Profilen von einem Mobilgerät auf ein anderes - werden vom Problem her ähnliche Anforderungen gestellt: Es gilt, all die einzelnen Angaben so gekennzeichnet abzulegen bzw. zu senden, dass später wieder die einzelnen Werte den richtigen Angaben im Profil zugewiesen werden können. Entsprechend ist der Aufbau der beiden sehr ähnlich.

Unterschiede zwischen den beiden liegen einerseits daran, dass teilweise nicht die gleiche Information übertragen wie gespeichert werden muss: Beispielsweise ist die Versionsnummer beim Speichern irrelevant, da der RecordStore zu der jeweiligen Applikationsversion fest dazugehört. Ausserdem muss ein "`NO_MATCH`" zum Beispiel zwar übertragen, jedoch nicht gespeichert werden können. Anders herum muss der "`FRIENDLY_NAME`" nicht explizit übertragen (er lässt sich direkt aus der Bluetooth-Verbindung auslesen), wohl aber gespeichert werden.
Andererseits erfordern auch strukturelle Unterschiede zwischen dem Abspeichrungsmechanismus und dem Übertragungsverfahren gewisse Anpassungen: Während bei der Übertragung jeweils ein Stream alle Daten der Übertragung fasst, war es beim Abspeichern naheliegend, ein RecordStore pro Profil jeweils mit einem Datensatz pro Eigenschaft zu füllen.
Die Abbildung 6.17 zeigt die Struktur eines Records des RecordStore zur Abspeicherung von Profilen, die Abbildung 6.18 veranschaulicht die Struktur der RecordStores.

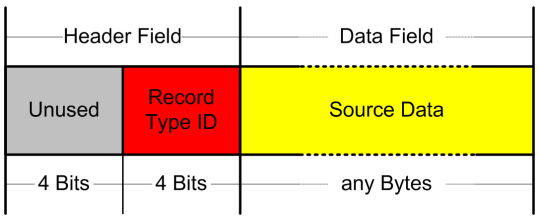


Abbildung 6.17: *Record Spezifikation*

Jeder RecordStore trägt einen eindeutigen Namen. Derjenige für das eigene Profil heisst "`ownProfile`", die RecordStores der von anderen Geräten empfangenen Profile haben als Namen die jeweilige Bluetooth-Adresse des Gerätes, welche in BlueDating als eindeutige Identifikation für ein anderes Bluetooth-Gerät dient.

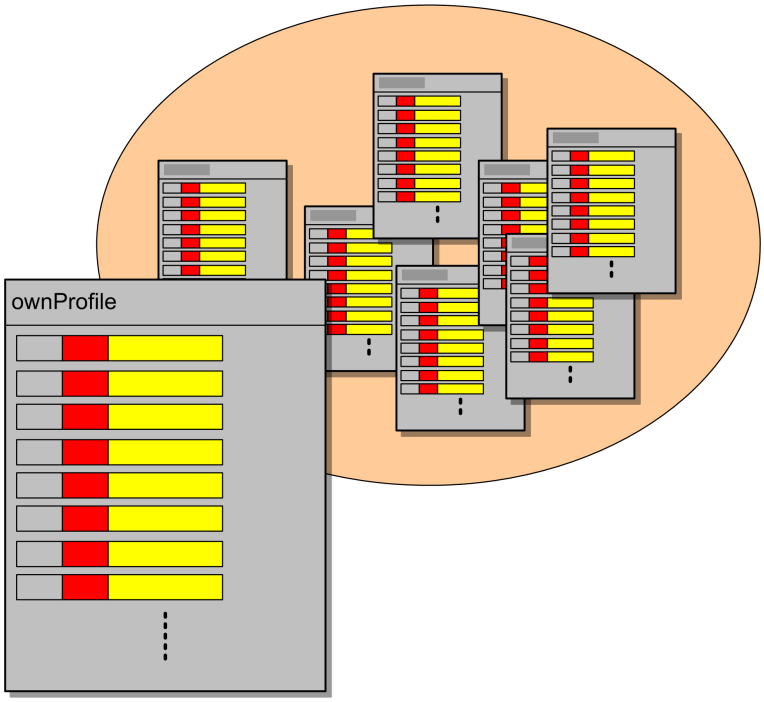


Abbildung 6.18: *Records der Profile in den RecordStores*

Die Möglichkeit, den RecordStore der Profile nur mit den jeweiligen Daten zu füllen und die Art der Daten über die jeweilige Position im RecordStore zu identifizieren, fehlt: Es kann nicht garantiert werden, dass die Reihenfolge erhalten bleibt (siehe Abschnitt 3.3).
Durch die Speicherung der Information im Record, zu welcher Profilangabe die Daten gehören, lässt sich dieses Problem umgehen. Die Reihenfolge der Records spielt nun keine Rolle mehr. Ausserdem muss so nicht zwingend das ganze Profil abgespeichert werden, denn beim Laden eines Profils wird erst ein Profil mit Standardwerten erstellt und dann werden die Daten beim Lesen des RecordStores angepasst. Fehlt eine Profilangabe im RecordStore, so bleibt der Standardwert erhalten.

6.7.3 Profile und andere zu speichernde Daten

Dieses Kapitel behandelt, wie Profile in RecordStores abgelegt werden. Neben dem Profil werden auch verschiedene Einstellungen und Zustände des BlueDating-Programms in RecordStores abgelegt. Da die Speicherung des Profils viele Gemeinsamkeiten mit dem Übertragungsprotokoll aufweist, werden die anderen zu speichernden Daten im Kapitel 6.8 behandelt. So sind dank der analogen Kapitelstruktur die Gemeinsamkeiten und Unterschiede der Speicherung und der Übertragung der Profile besser ersichtlich.

6.7.4 Das Record Type Byte

Dies ist mit dem Kopf des Übertragungsprotokolls vergleichbar. Da nur jeweils ein Profileintrag pro Record gespeichert wird erübrigt es sich, hier eine Länge anzugeben. Auch die Versionsangabe entfällt, da die Records sowieso direkt mit dem Programm verknüpft sind und nur von diesem gelesen werden können.

6.7.4.1 Die Record Type ID

Die Record Type ID entspricht weitgehend der Message Type ID des Übertragungsprotokolls, sie ist deshalb ebenfalls 4 Bits lang. Die Tabelle 6.9 beschreibt die verschiedenen Message Type IDs.

6.7.4.2 Unbenutzte Bits

Da die Record Type ID auf 4 Bits begrenzt bleibt, verbleiben 4 unbenutzte Bits im Record Type Byte.

6.7.5 Das Datenfeld

6.7.5.1 Das Datenfeld von "Gesuchte Core Characteristics"

Die Abbildung 6.19 zeigt das Datenfeld von "Gesuchte Core Characteristics".

6.7.5.1.1 Die Attribute Type ID

Die Tabelle 6.10 beschreibt die verschiedenen Attribute Type IDs von "Gesuchte Core Characteristics".

Record Type ID [hex]	Bezeichnung	Beschreibung
0x0	SEARCH_CORE	Gesuchte Basiseigenschaften
0x1	SEARCH_EXT	Gesuchte erweiterte Eigenschaften
0x2	OWN_CORE	Eigene Basiseigenschaften
0x3	OWN_EXT	Eigene erweiterte Eigenschaften
0x4	ADD_INFO	Zusätzliche Informationen
0xE	ADDITIONAL_PHONE_INFO	"Zusätzliche Telefon Informationen": Informationen über das Telefon mit dem man verbunden ist
0xF	PARTS_COMPLETED	Informationen, welche Teile der Profile ausgefüllt sind oder von einem anderen Telefon erhalten worden sind

Tabelle 6.9: *Record Type ID*

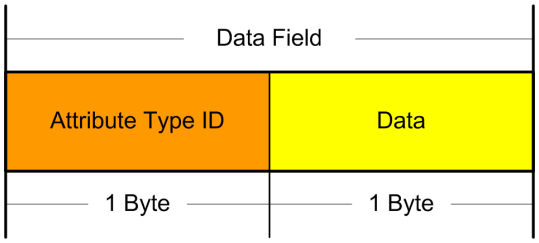


Abbildung 6.19: *Das Datenfeld von "Gesuchte Core Characteristics"*

Attribute Type ID [hex]	Bezeichnung	Beschreibung
0x00	SEX	Geschlecht
0x01	AGE_FROM	Alter ab
0x02	AGE_TO	Alter bis
0x03	HEIGHT_FROM	Grösse ab
0x04	HEIGHT_TO	Grösse bis
0x05	WEIGHT_FROM	Gewicht ab
0x06	WEIGHT_TO	Gewicht bis
0x07	HAIR	Haarfarbe
0x08	EYE	Augenfarbe
0x09	SMOKE	Rauchgewohnheiten
0x0A	DRINK	Trinkgewohnheiten
0x0B	RELATIONSHIP	Beziehungsart

Tabelle 6.10: *Attribute Type IDs von "Gesuchte Core Characteristics"*

6.7.5.1.2 Die Daten

Das Datenbyte kann auf zwei verschiedene Arten aufgebaut sein, was abhängig von der Attribute Type ID ist. Entweder bitweise oder das ganze Byte als einen unsigned Interger.

Bitweise

Dies wird bei Eigenschaften gebraucht, welche eine Auswahl an verschiedenen Kriterien zulassen. Jedem Kriterium wird ein Bit zugeordnet, welchesn auf 1 gesetzt werden kann, wenn es zutrifft. Falls der Benutzer keine Eigenschaft auswählt, steht das Byte auf `0xFF`, was als "Egal" interpretiert wird. Folgende Attribute Type IDs verwenden die bitweise Darstellung:

Geschlecht
Die Abbildung 6.20 zeigt die Daten von "Geschlecht".

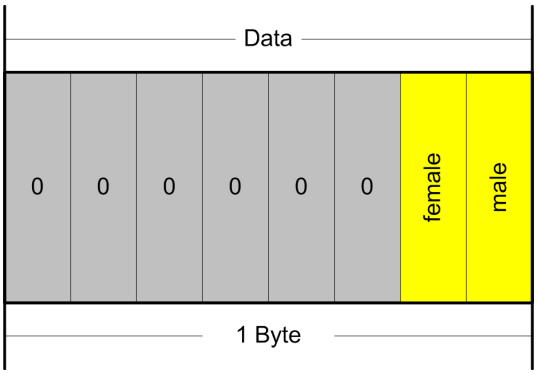


Abbildung 6.20: *Die Daten von "Geschlecht"*

Haarfarbe
Die Abbildung 6.21 zeigt die Daten von "Geschlecht".

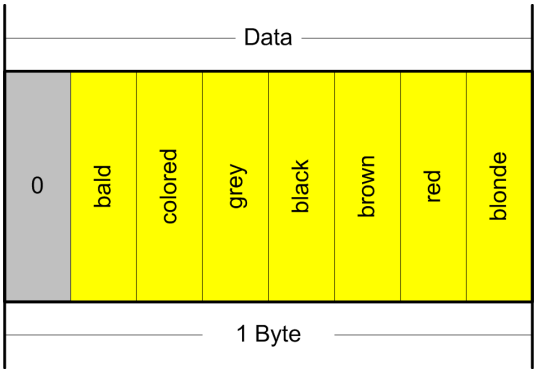


Abbildung 6.21: *Die Daten von "Haarfarbe"*

Augenfarbe
Die Abbildung 6.22 zeigt die Daten von "Augenfarbe".

Rauchgewohnheiten

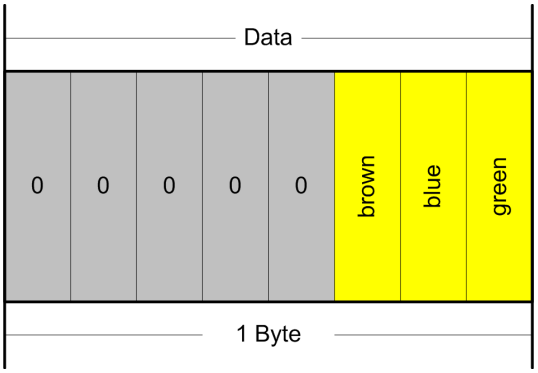


Abbildung 6.22: *Die Daten von "Augenfarbe"*

Die Abbildung 6.23 zeigt die Daten von "Rauchgewohnheiten".

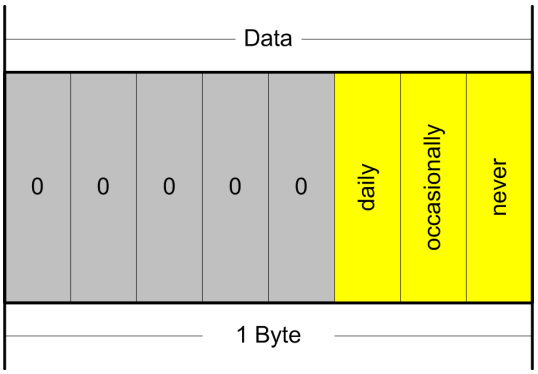


Abbildung 6.23: *Die Daten von "Rauchgewohnheiten"*

Trinkgewohnheiten

Die Abbildung 6.24 zeigt die Daten von "Trinkgewohnheiten".

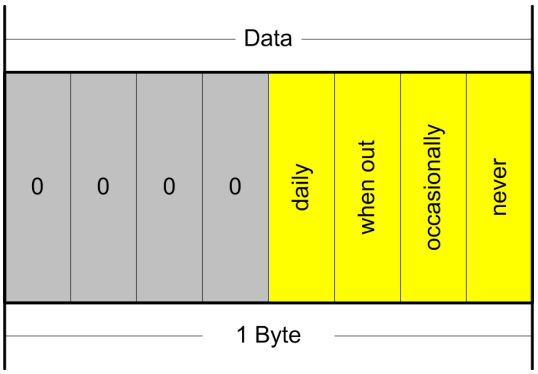


Abbildung 6.24: *Die Daten von "Trinkgewohnheiten"*

Beziehungsart

Die Abbildung 6.25 zeigt die Daten von "Beziehungsart".

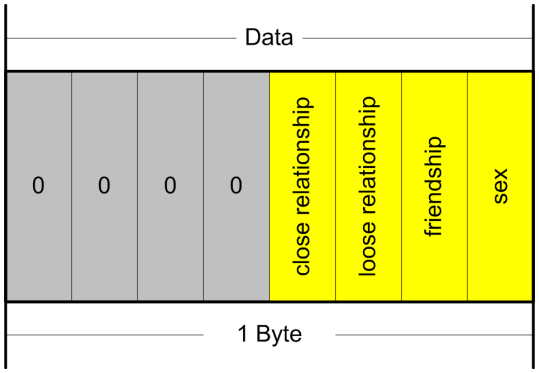


Abbildung 6.25: *Die Daten von "Beziehungsart"*

Byteweise

Das Datenbyte wird als ganzes Byte gesetzt, das heisst es wird einfach der Wert als `8 Bit unsigned Integer` geschrieben. Falls der Benutzer die Auswahl "Egal" trifft, wird das Byte auf `0x00` gesetzt. Folgende Attribute Type IDs verwenden die byteweise Darstellung:

- Age from/to
- Height from/to
- Weight from/to

Es sind Werte im Bereich `0x01` bis `0xFE` (254) erlaubt, da der Wert `0x00` für die "Egal"-Auswahl reserviert ist.

6.7.5.2 Das Datenfeld von "Gesuchte Extended Characteristics"

Das Datenfeld von "Gesuchte Extended Characteristics" hat die gleiche Struktur wie das Datenfeld von "Gesuchte Core Characteristics" in Abbildung 6.19.

6.7.5.2.1 Die Attribute Type ID

Die Tabelle 6.11 beschreibt die verschiedenen Attribute Type IDs von "Gesuchte Extended Characteristics".

6.7.5.2.2 Die Daten

Alle Daten werden byteweise geschrieben, da ja die Auswahl des Benutzers ein Wert zwischen 0 und 5 ist. 0 steht dabei für "Egal".

6.7.5.3 Das Datenfeld von "Eigene Core Characteristics"

Das Datenfeld von "Eigene Core Characteristics" hat die gleiche Struktur wie das Datenfeld von "Gesuchte Core Characteristics" in Abbildung 6.19.

6.7.5.3.1 Die Attribute Type ID

Die Tabelle 6.12 beschreibt die verschiedenen Attribute Type IDs von "Eigene Core Characteristics".

Attribute Type ID [hex]	Bezeichnung	Beschreibung
0x00	SPORTS	Bewegungsfaul / Sportfanatiker
0x01	BOOKS_TV	Lesen / Kino, TV
0x02	PARTY_RELAX	Party, Ausgang / Ruhiger Abend
0x03	TRAMP_HOTEL	Trampen / 5-Sterne-Hotel
0x04	NIGHT_DAY	Nachtmensch / Früh ins Bett
0x05	MORNING	Frühaufsteher / Morgenmuffel
0x06	HARMONY	Harmonie / Energie
0x07	ARMANI	Armani / 2nd Hand
0x08	PEN_COMP	Papier und Bleistift / Computerfreak
0x09	FAST_REST	Take Out, Fast Food / Restaurant
0x0A	WORLD_LOCAL	Weltentdecker / Stubenhocker
0x0B	CITY	Stadtmensch / Landmensch
0x0C	ZOO	Kein Haustier / "Zoo" daheim
0x0D	IN_OUTDOOR	Indoor / Outdoor
0x0E	GROUP	Einzelgänger / Gruppenmensch

Tabelle 6.11: *Attribute Type IDs von "Gesuchte Extended Characteristics"*

Attribute Type ID [hex]	Bezeichnung	Beschreibung
0x00	SEX	Geschlecht
0x01	AGE	Alter
0x03	HEIGHT	Grösse
0x05	WEIGHT	Gewicht
0x07	HAIR	Haarfarbe
0x08	EYE	Augenfarbe
0x09	SMOKE	Rauchgewohnheiten
0x0A	DRINK	Trinkgewohnheiten

Tabelle 6.12: *Attribute Type IDs von "Eigene Core Characteristics"*

6.7.5.3.2 Die Daten

Das Datenbyte kann auf zwei verschiedene Arten aufgebaut sein, was abhängig von der Attribute Type ID ist. Entweder bitweise oder das ganze Byte als einen unsigned Interger.

Bitweise

Dies wird bei gebraucht, welche eine Auswahl an verschiedenen Kriterien zulassen. Jedem Kriterium wird ein Bit zugeordnet, welches auf 1 gesetzt werden kann, wenn es zutrifft. Die Daten haben die gleiche Struktur wie im Abschnitt 6.7.5.1.2, jedoch ohne die Möglichkeit den Wert für "Egal" zu wählen.

Byteweise

Das Datenbyte wird als ganzes Byte gesetzt, das heisst es wird einfach der Wert als `8 Bit unsigned Integer` geschrieben. Folgende Attribute Type IDs verwenden die byteweise Darstellung:

- Age
- Height
- Weight

Es sind Werte im Bereich `0x0` bis `0xFE` (254) erlaubt.

6.7.5.4 Das Datenfeld von "Eigene Extended Characteristics"

Das Datenfeld von "Eigene Extended Characteristics" sieht gleich aus wie das Datenfeld von "Sende gesuchte Extended Characteristics" im Abschnitt 6.7.5.2, mit der Ausnahme, dass 0 nicht für "Egal" sondern für "Keine Angabe" steht.

6.7.5.5 Das Datenfeld von "Additional Information"

Die Abbildung 6.26 zeigt das Datenfeld von "Sende Additional Information".

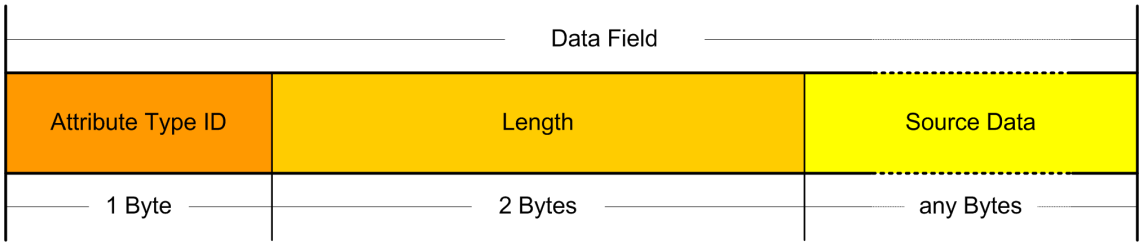


Abbildung 6.26: *Das Datenfeld von "Additional Information"*

6.7.5.5.1 Die Attribute Type ID

Die Tabelle 6.13 beschreibt die verschiedenen Attribute Type IDs von "Additional Information".

Attribute Type ID [hex]	Bezeichnung	Beschreibung
`0x00`	HANDY	Nummer des Mobiltelefons
`0x01`	EMAIL	E-mail Adresse
`0x02`	COMMENT	Kommentar

Tabelle 6.13: *Attribute Type IDs von "Additional Information"*

6.7.5.5.2 Die Länge

Da bei den Einträgen der "Additional Information" die Angaben in der Länge variieren, enthalten diese 2 Bytes die Länge des nachfolgenden Datenblocks. Mit 2 Bytes sind Datenblocklängen bis zu 65 kB möglich.

6.7.5.5.3 Die Daten

Die Werte werden Byteweise geschrieben. Ein Byte entspricht einem Zeichen (CHAR).

6.7.5.6 Das Datenfeld von "Additional Phone Information"

Unter diesem Attributs-Typ "Additional Phone Information" werden zusätzliche Angaben zum Mobilgerät, zu welchem das Profil gehört, abgelegt. Es handelt sich um Informationen, welche vom anderen Mobilgerät im Rahmen der Verbindung automatisch mitübertragen werden. Bislang ist dies nur der Eintrag "Friendly Name", wie aus der Tabelle 6.14 hervorgeht.

Attribute Type ID [hex]	Bezeichnung	Beschreibung
0x00	FRIENDLY_NAME	Der Bluetooth-Name

Tabelle 6.14: *Attribute Type IDs von "Additional Information"*

6.7.5.7 Das Datenfeld von "Parts Completed"

Das Datenfeld für die Message Type ID "Parts Completed" gibt Auskunft darüber, ob die Daten eines Teils des Profils bereits übertragen bzw. ausgefüllt wurden. Es hat den selben Aufbau wie bei das Datenfeld der "Core Characteristics", also wie in Abbildung 6.19 aufgezeigt.

6.7.5.7.1 Die Attribute Type ID

Die Tabelle 6.15 beschreibt die verschiedenen Attribute Type IDs von "Parts Completed".

Attribute Type ID [hex]	Bezeichnung	Beschreibung
0x00	COMPLETED_SEARCH_CORE	Die gesuchten Basiseigenschaften sind komplett
0x01	COMPLETED_SEARCH_EXT	Die gesuchten erweiterten Eigenschaften sind komplett
0x02	COMPLETED_OWN_CORE	Die eigenen Basiseigenschaften sind komplett
0x03	COMPLETED_OWN_EXT	Das eigenen erweiterte Eigenschaften sind komplett

Tabelle 6.15: *Attribute Type IDs von "Additional Information"*

6.7.5.7.2 Die Daten

Das nachfolgende Byte enthält den Wert 1, falls der entsprechende Teil des Profils bereits gefüllt wurde und 0, falls dies noch nicht geschehen ist.

6.8 Einstellungs- und Zustands-Abspeicherungs-Spezifikation

6.8.1 Übersicht

Neben den Profilen werden folgende Daten dauerhaft in RecordStores abgelegt:

- KnownDevices
- LastMatches
- Settings
- Statistics

Für jeden der obengenannten Punkte wird ein eigener RecordStore mit entsprechendem Namen angelegt.

6.8.2 KnownDevices

Die LastMatches-Liste umfasst mehrere Datensätze, die in der Tabelle 6.16 ersichtlich sind.

Datentyp	Bezeichnung	Beschreibung
Long	Calendar	Datum und Uhrzeit des Eintrags
Unsigned Byte	StringLength	Länge der Bluetooth-Adresse des anderen Geräts
String	DeviceBluetoothAddress	Die Bluetooth-Adresse des anderen Geräts

Tabelle 6.16: *Record Type ID*

Die KnownDevices-Liste wird im RecordStore "`KnownDevices`" abgespeichert. Pro Datensatz wird ein Record erstellt, dessen Gliederung in der Abbildung 6.27 aufgezeigt ist.

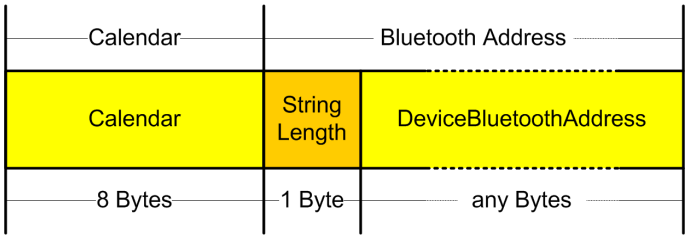


Abbildung 6.27: *Record von KnownDevices*

6.8.3 LastMatches

Die LastMatches-Liste umfasst mehrere Datensätze, die in der Tabelle 6.17 ersichtlich sind.
Die LastMatches-Liste wird im RecordStore "`LastMatches`" abgespeichert. Pro Datensatz wird ein Record erstellt, dessen Gliederung in der Abbildung 6.28 aufgezeigt ist.

Datentyp	Bezeichnung	Beschreibung
Long	Calendar	Datum und Uhrzeit des Eintrags
Unsigned Byte	NewFlag	1 falls Neu, 0 andernfalls
Unsigned Byte	StringLength	Länge der Bluetooth-Adresse des anderen Geräts
String	DeviceBluetoothAddress	Die Bluetooth-Adresse des anderen Geräts

Tabelle 6.17: *Record Type ID*

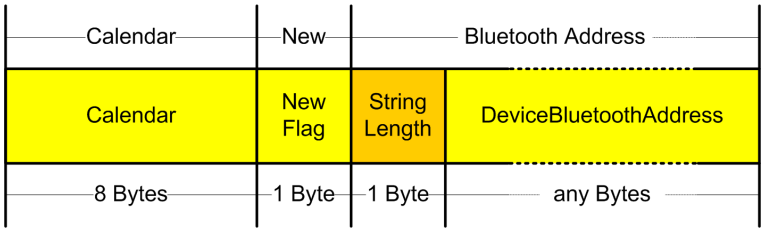


Abbildung 6.28: *Record von LastMatches*

6.8.4 Settings

Die vom Benutzer gewählten Einstellungen werden im RecordStore "`SettingsRecords`" gespeichert. Dabei handelt es sich um zwei Integer-Werte:

- Such-Intervall
- Such-Modus

Diese werden je in einen Record des RecordStores gespeichert, wie Abbildung 6.29 veranschaulicht: Die 4 Bytes langen Daten folgen auf ein Headerbyte in welchem steht, welcher Wert im entsprechenden Record gespeichert ist. Die Werte, welche das Headerbyte annehmen kann, sind in Tabelle 6.18 beschrieben.

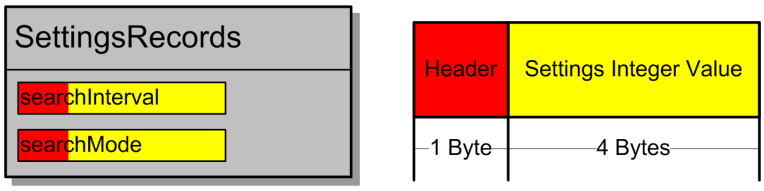


Abbildung 6.29: *RecordStore und Record der Settings*

Header Byte	Bezeichnung	Beschreibung
`0x00`	SETTINGS_INTERVAL	Das Such-Intervall
`0x01`	SETTINGS_MODE	Der Such-Modus

Tabelle 6.18: *Record Type ID*

6.8.5 Statistics

Die Statistik-Werte des BlueDating-Programms werden im RecordStore "`StatisticRecords`" gespeichert. Dabei handelt es sich um drei Integer-Werte:

- Anzahl der Verbindungen zu anderen BlueDating-Programmen
- Anzahl der Matches
- Anzahl der Suchgänge

Analog zu den Einstellungen werden jeweils ein Headerbyte und vier Datenbyte, wie in Abbildung 6.30 dargestellt, in einen Record gespeichert. Die Tabelle 6.19 gibt über die jeweiligen Headerbytes Auskunft.

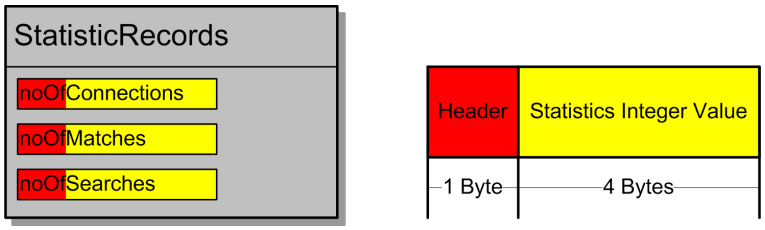


Abbildung 6.30: *RecordStore und Record der Settings*

Header Byte	Bezeichnung	Beschreibung
`0x00`	NO_OF_CONNECTIONS	Anzahl der Verbindungen zu anderen BlueDating-Programmen
`0x01`	NO_OF_MATCHES	Anzahl der Matches
`0x02`	NO_OF_SEARCHES	Anzahl der Suchgänge

Tabelle 6.19: *Record Type ID*

6.9 Semaphore

Semaphore dienen dem Zweck, sogenannten "kritischen Code" gegen aussen abzusichern. Das bedeutet, dass höchstens ein Thread aufs Mal diesen Code ausführen darf und alle anderen müssen sich hinten anstellen. Ist dieser Thread fertig, bekommt ein anderer das Recht diesen Code auszuführen. Solche kritischen Code-Segmente sind zum Beispiel Diskzugriffe oder andere I/O Aktivitäten.
Bei der Implementation von BlueDating wurden zwei Semaphoren eingesetzt, der `deviceListSemaphore` und der `dataAccessSemaphore`.

6.9.1 Der DataAccessSemaphore

Dieser dient dem Zweck, dass immer nur ein bestimmter Thread RecordStores lesen oder auch schreiben kann. Dies aus dem Grund, da immer auch Längenangaben zum lesen oder schreiben eines RecordStores gehören. Diese Längen müssen errechnet oder ausgelesen werden, bevor der eigentliche Lese/Schreib-Zugriff stattfindet.

Ein anderer Thread könnte, nachdem die Länge erkundet wurde, aber noch bevor der Zugriff stattfindet, den RecordStore ändern, ja gar löschen. Dies könnte im schlimmsten Falle falsches Programmverhalten hervorrufen oder das Programm gar zum Absturz bringen. Da mobile Geräte nicht so flink sind, vergeht auch seine Zeit bis ein Zugriff erfolgreich beendet ist. Während dem ganzen Zugriffzyklus darf daher immer nur ein Thread die RecordStores manipulieren.

6.9.2 Der DevicelistSemaphore

Etwas Ähnliches gilt bei den beiden globalen Vektoren `knownDevices` und `lastMatches`, die Funktionen zum Auslesen der Grösse, für das Lesen und das Schreiben zulassen. Liest man nun die Länge des Vektors aus, und ein anderer löscht kurz danach ein Element, könnte eine Schlaufe mit der vorherausgelesenen Länge grosse Probleme verursachen.
Daher wurden die Lese- und Schreibzugriffe auf diese beiden Vektoren durch einen Semaphoren gegen solche Probleme geschützt.

Kapitel 7

Schlussfolgerungen

7.1 Resultate

Mit BlueDating ist eine Partnersuche mit javafähigen Mobiltelefonen über Bluetooth möglich. Dies war das Hauptziel unserer Aufgabenstellung.
Eine wichtige Erkenntnis, die wir bei dieser Arbeit wieder einmal hautnah erkennen mussten, ist der Unterschied eines Emulators gegenüber dem richtigen Gerät. Im Emulator hat man mit der Zeitdauer der Ausführung kein Problem, da die CPU genügend schnell rechnet. Ebenfalls Speicher hat man im Emulator genug.
Eine Software für ein eingebettetes System zu entwickeln, heisst automatisch auch ressourcenminimiert zu programmieren. Je weniger Speicher oder je weniger Zugriffe auf die RecordStores man braucht, desto besser und schneller läuft die Software.
Wie im Abschnitt 4.3 genauer erklärt, kostete es erheblichen Aufwand, die bekannten Fehler in der Implementation von JSR-82 im Nokia 6630 zu umgehen.
Bis zum Schluss bereitete uns die Trägheit der graphischen Oberfläche in Java erhebliche Mühe. Pop-Ups kommen nie dann, wenn man sie möchte. Sogar die Reihenfolge von mehreren Pop-Ups kann ändern.

7.2 Ausblick

Unsere Applikation könnte kommerzialisiert werden. Das komplette Konzept ist so ausgelegt, dass dies mit relativ geringem Aufwand realisierbar ist.
Weiter Anwendungsmöglichkeiten könnten wie in Abschnitt 5 genauer erwähnt, beispielsweise ein System für Kongresse oder gar eine Hotelzimmersuche sein.

7.3 Danksagung

Wir möchten uns an dieser Stelle ganz herzlich bei unseren Betreuern, Matthias Bosshardt, Vincent Lenders und Dr. Martin May für deren freundliche Unterstützung bedanken. Das angenehme Arbeitsklima hat unsere Kreativität beflügelt.

Anhang A

Eingebaute Debug Funktionalität

Bei der Entwicklung ist es manchmal vonnöten, interne Variablen und Zustände des Programms beeinflussen zu können, um bestimmte Situationen durchzuspielen. Ebenso ist dies für Demonstrationszwecke wünschenswert. Das "Debug-Menü", falls eingeschaltet erreichbar über einen Punkt im Hauptmenü von BlueDating, erfüllt dieses Bedürfnis.

A.1 Funktionen des Debug-Menüs

A.1.1 Eigenes Profil wechseln

Zwar kann das eigene Profil über den Punkt "Profil editieren" im Hauptmenü eingegeben und modifiziert werden, diese Methode ist bei häufigem Wechsel des Profils jedoch unpraktisch, da zeitaufwändig.
In BlueDating sind sechs Profile zu Test- und Demonstrationszwecken vordefiniert. Diese können über den Punkt "Eigenes Profil wechseln" des Debug-Menüs geladen werden.
Tabelle A.1 zeigt auf, welche Debug-Profile mit welchen matchen (✓: Match, **G**: Match nur in Grundeigenschaften, ✗: kein Match). Die Profile 1-3 sind allgemeine Testprofile, 4-6 sind auf die Demonstration zugeschnitten: 4 ist das Profil von Ken, 5 jenes von Barbie, welches mit Ken einen Match ergibt. Das Profil 6 gehört zu Kens Zwillingsbruder, welcher die selben Grundeigenschaften mitbringt wie Ken, jedoch charakterlich (also in den erweiterten Eigenschaften) sich von Ken so stark unterscheidet, dass er nicht mehr Barbies Typ ist.

Profil	1	2	3	4	5	6
1	✓	✗	✗	✗	✗	✗
2	✗	✗	✗	✗	✗	✗
3	✗	✗	✗	✗	✗	✗
4	✗	✗	✗	✗	✓	✗
5	✗	✗	✗	✓	✗	**G**
6	✗	✗	✗	✗	**G**	✗

Tabelle A.1: *Matches zwischen den Debug-Profilen*

A.1.2 Letzte Matches füllen

Um die Funktionsweise des "Letzte Matches"-Menüs zu demonstrieren kann es unter "Letzte Matches füllen" im Debug-Menü mit drei Einträgen gefüllt werden. Dabei kommen als Matches die Debugprofile 1-3 zum Einsatz. Dass diese mit dem aktuellen eigenen Profil wohl gar keinen Match ergäben spielt in diesem Zusammenhang keine Rolle.

A.1.3 Freien Speicher anzeigen

Der momentan freie Arbeitsspeicher des Mobilgeräts kann mit dem Menüpunkt "Freien Speicher anzeigen" auf einem Popup-Fenster angezeigt werden.

A.1.4 Vektoren und Profile entleeren

Die *Known Devices*- und die *Last Matches*-Liste, sowie zugehörige Profile werden gelöscht, wenn im Debug-Menü "Vektoren und Profile entleeren" gewählt wird.
Dies ist insbesondere nötig, damit zwei Geräte, die eben erst einen Match hatten, wieder überhaupt versuchen, einen Match zu erreichen und nicht schon abbrechen, wenn sie das jeweils andere Gerät als bekannt in der entsprechenden Liste finden.

A.2 Aktivierung des Debug-Menüs

Das Debug-Menü ist in der Laborumgebung aktiv, sollte jedoch deaktiviert werden, wenn es normalen Benutzern abgegeben wird. Dies geschieht im Javacode, indem in der Klasse BlueDateGUI die Zeile

```
MainMenu.append(''DEBUG MENU'', null);
```

auskommentiert wird.

Anhang B

Entwicklungsumgebung

Dieses Kapitel vermittelt eine Übersicht über die genutzen Programme. Es wird anschliessend darauf eingegangen, wie man alle notwendigen Programme für *BlueDating* installiert und wie *BlueDating* auf einem Emulator ausgeführt wird. Ebenfalls wird die Installation auf ein reales Gerät, dem Nokia 6630 (siehe B.3), beschrieben.

B.1 Sun Java Studio Mobility

Sun Java Studio Mobility in der Version 6 2004Q3 ist kostenlos. Dies war der Hauptgrund warum wir dieses Produkt vor kostenpflichtigen wie z.B.: jBuilder von Borland bevorzugten.
Die Abbildungen B.2 zeigt diese Entwicklungsumgebung.

Abbildung B.1: *Sun Java Studio Mobility 6 2004Q3*

Sun Java Studio Mobility ist eine moderne Entwicklungsumgebung welche für die Entwicklung von Applikationen auf mobilen Geräten, die Java Technologie unterstützen, optimiert. Es benutzt die Java 2 Micro Edition CLDC/MIDP Plattform. Für uns war es sehr angenehm mit dieser Umgebung zu programmieren, da sie einem viel Arbeit abnimmt. Neue Klassen lassen sich als leeres Gerüst erstellen und die Java Dokumentation, die man auf einem Schlüsselwort aufrufen kann, ist sehr hilfreich. Man kann Sun eigene Emulatoren (siehe B.2.2) oder auch fremde, wie den Nokia Emulator (siehe B.2.1), direkt einbetten.

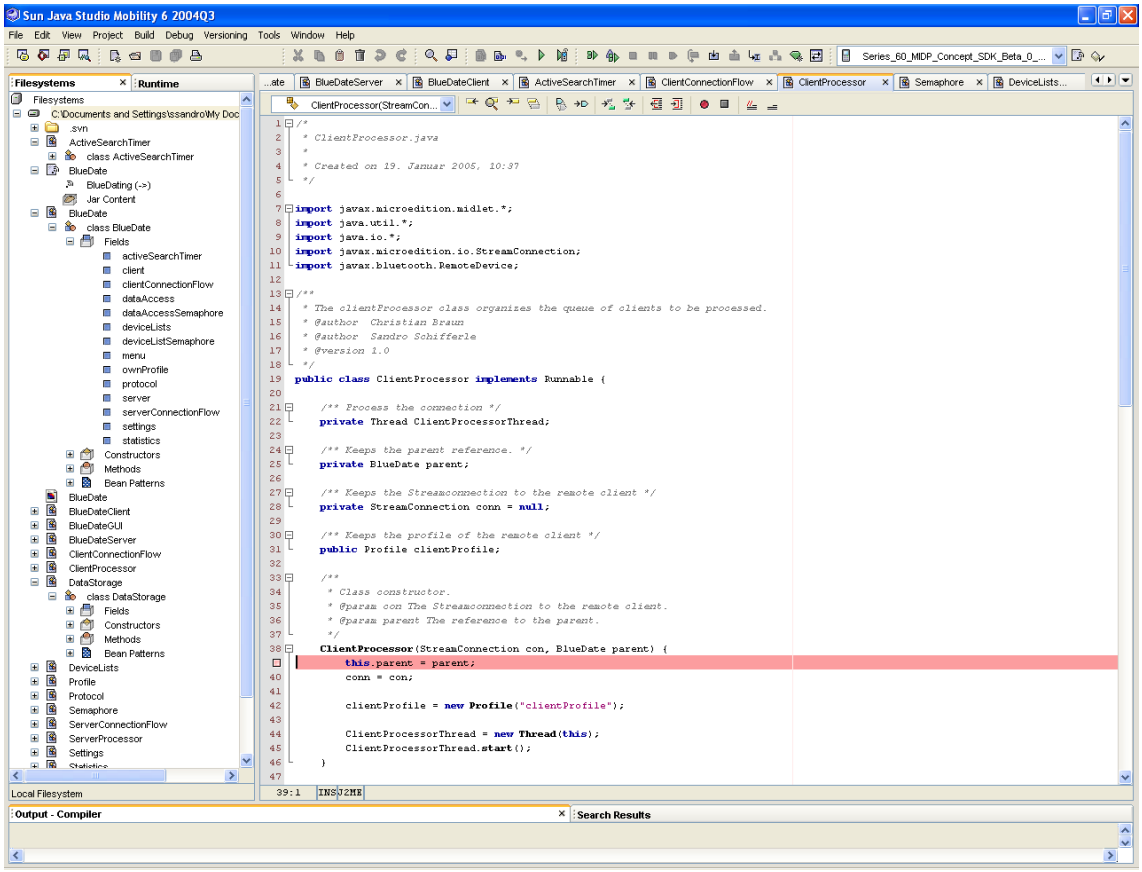


Abbildung B.2: *Sun Java Studio Mobility 6 2004Q3*

Leider hat sie auch seine Tücken. Sie hat Probleme mit Breakpoints beim Debuggen, wenn mehrere Threads aktiv sind und sich in diesem Breakpoint gefinden. Sie kann danach jeweils nur noch einen wieder zum laufen bringen. Ebenfalls hat sie Mühe mit Breakpoints in Klassenkonstruktoren.
Die Software kann unter

`http://developers.sun.com/prodtech/javatools/jsmobility/downloads/index.html`

heruntergeladen werden. Man muss sich dazu jedoch vorweg bei Sun registrieren, was ebenfalls kostenlos ist. Die Installation ist unter B.4.1.1 beschrieben.

B.2 Emulatoren

Um auf das Bluetooth API JSR-82 oder auf das Dateisystem von Mobiltelefonen zuzugreifen können, braucht man einen Emulator, der dies unterstützt. Es gibt von fast jedem Mobiltelefonhersteller spezifische Emulatoren. Wir haben hauptsächlich die zwei folgenden genutzt:

B.2.1 Nokia Developer's Suite for J2ME

Die Noka Developer's Suite beinhaltet eine komplette Entwicklungsumgebung. Wir benutzten die Version 2.2, welche man herunterladen kann, unter:

`http://www.forum.nokia.com/main/0,6566,034-2,00.html`

Zusätzlich braucht man noch das "Series 60 MIDP Concept SDK Beta 0.3.1", welches einem den Zugriff auf das Bluetooth API JSR-82 ermöglicht und zugleich als ein Update gilt. Dieses bekommt man unter:

`http://www.forum.nokia.com/main/1,6566,034-243,00.html`

Für beide Tools benötigt man von Nokia einen Key, welchen man sich durch Gratisregistrierung zukommen lassen kann.
Der Nokia Emulator eignet sich vor allem um zu sehen, wie die realen Nokia Geräte sich verhalten. Man kann die Speicher, Geschwindigkeit und Grafikverzögerung ebenfalls emulieren.
Leider wird mit dem Nokia 6230 das einzige reale Gerät emuliert. Für unser Nokia 6630 musste man sich doch auf dem realen Gerät vergewissern, dass es funktioniert.
B.3 zeigt die verschieden Geräte welche der Nokia Emulator der Serie 60 emulieren kann.
Die Installation ist unter B.4.1.2 beschrieben.

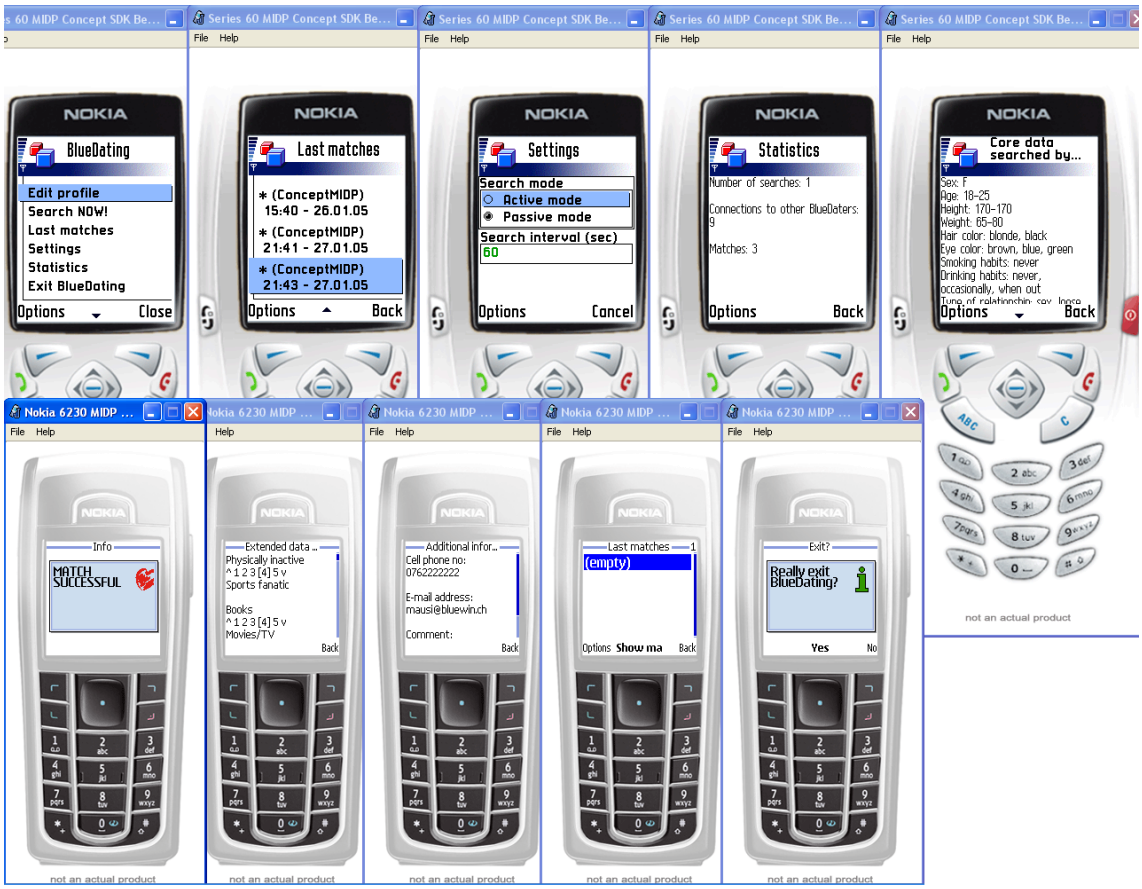


Abbildung B.3: *Nokia Emulator*

B.2.2 J2ME Wireless Toolkit

Das J2ME Wireless Toolkit 2.2 ist die SUN-Variante eines Emulators. Das Toolkit hat den Vorteil, dass der Emulator nicht an reale Geräte gebunden ist, auf denen

es dann genauso aussehen und funktioneren sollte, da SUN ja keine eigenen Mobiltelefone herstellt. Daher ist dieser Emulator sehr allgemein, unterstützt alle bis zu diesem Zeitpunkt definierten API's und ist relativ stabil.
Ebenfalls besteht die Möglichkeit einen "Memory monitor" oder "Network monitor" zu aktivieren, welche einem die genaue Übersicht über das Verhalten der Software gibt.
B.4 zeigt die verschieden Geräte welche das J2ME Wireless Toolkit 2.2 emulieren kann.

Abbildung B.4: *J2ME Wireless Toolkit*

Das Toolkit kann unter

`http://java.sun.com/products/j2mewtoolkit/download-2_2.html`

heruntergeladen werden.
Die Installation ist unter B.4.1.3 beschrieben.

B.3 Nokia 6630

Das Nokia 6630 kam Ende 2004 auf den Markt. Es ist das erste UMTS Mobiltelefon, das auf dem Markt erhältlich ist. Die Funktionalität dieses Gerätes ist enorm, siehe auch unter [12]. Wir haben uns für dieses Modell entschieden, weil es das erste Mobiltelefon von Nokia ist, welches das JSR-82 API (Bluetooth) und das JSR-75 API (Dateisystem) zur Verfügung stellt.

Leider mussten wir feststellen, dass die Firmware dieses Gerätes überhaupt noch nicht ausgereift ist und das Gerät zu diversen und jeweils verschiedenen Zeitpunkten einfach abstürzt. Nokia bestätigt auf seiner Entwicklerseite [13] einige Fehler im Zusammenhang mit diesen Phänomenen. Ebenfalls das Bluetooth scheint noch einige Fehler zu enthalten, die es momentan fast unmöglich machen, zwei Verbindungen zu je einem anderen Gerät gleichzeitig offen zu halten.

Abbildung B.5: *Das Nokia 6630*

Falls es zu einem Programmabsturz kommen sollte, kann man das Nokia 6630 mit folgender Eingabe zurücksetzen: `*#7780#`. Sollte dies nicht reichen, ist es auch möglich, einen *Reset to Factory Default* durchzuführen. Mann muss dazu die grüne Taste zusammen mit `3` und `*` gedrückt halten, während man das Gerät einschaltet, solange, bis man wieder, wie beim ersten mal starten, das Land wählen kann, in dem man sich befindet (Diese zweite Variante ist laut Nokia Kundenservice ohne Garantie).

B.4 Installation und Ausführen von BlueDating

Dieser Abschnitt erklärt, wie die vorgestellte Software zu installieren ist und wie man damit *BlueDating* ausführen kann.

B.4.1 Software Installation

Systemvoraussetzung ist ein PC mit Windows 2000 oder XP. Ein starke CPU und viel RAM ist von Vorteil, da das Emulieren stark an diesen Ressourcen zerrt.
Folgende Softwarepakete müssen installiert werden:

- Sun Java Studio Mobility 6 2004Q3
- Entweder den Nokia Emulator, den Sun Emulator oder die Nokia PC Suite für das 6630

B.4.1.1 Installation des Sun Java Studio Mobility 6 2004Q3

Wie schon angetönt, benötigt man von Sun dazu einen gültigen Schlüssel. Um das Sun Java Studio Mobility 6 2004Q3 zu installieren ist das Java 2 SDK notwendig, welches man unter `http://java.sun.com/j2se/` bekommt. Es empfielt eine allfällig vorhandene ältere Version des Java 2 SDK vorher zu deinstallieren, um Versionskonflikte zu vermeiden.

B.4.1.2 Installation des Nokia Emulators

Die Software (siehe B.3) muss mit dem Nokia Schlüssel installiert werden. Es besteht die Möglichkeit, den Emulator standalone zu benutzen, dazu braucht man jedoch schon eine kompilierte Version des Java-Programms. Um den Emulator im Sun JAVA Mobility Studio zu integieren, muss man unter dem Register `Runtime` unter `Device Emulator Registry` den Emulator hinzufügen. Nun kann man im Toolmenu diesen Emulator auswählen.

B.4.1.3 Installation des Sun Emulators

Die Installation des Sun Wireless Toolkits ist das einfachste am ganzen. Doppelklicken, warten, fertig.

B.4.1.4 Installation der Nokia PC-Suite

Um die Nokia PC-Suite auf dem eigenen Computer zum laufen zu bringen, probiert man am besten die mit dem Nokia 6630 mitgelieferte Version. Falls diese nicht funktionieren sollte (was bei uns der Fall war), muss man von der Nokia Seite eine aktuelle Version herunterladen.

B.4.2 Kompilierung und Ausführung

Dieser Abschnitt beschreibt, wie man *BlueDating* zum laufen bringt. Hat man schon eine kompilierte Version (ein `BlueDating.jar` und `BlueDating.jad`), kann man diese Dateien direkt im standalone Emulator laden und testen. Der Nokia und der Sun Emulator bieten diese Möglichkeit an.

B.4.2.1 Direkt ab CD kompilieren

Dies ist die schnellstmögliche Variante den Code zu kompilieren. Als Software braucht man dazu lediglich das J2ME Wireless Toolkit (siehe Abschnitt B.2.2). Man starte das Tool *KToolbar*. Man kann das Projekt jetzt öffnen, wenn man das komplette Verzeichnis `programm\BlueDate\` der CD-ROM nach `C:\WTK22\apps\` kopiert, da *KToolbar* nur von hier Projekte öffnen kann (in seinem Installationsverzeichnis). Mit dem Knopf *Build* wird das Projekt kompiliert. Mit dem Knopf *Run* kann man es im Emulator ausführen.

B.4.2.2 Mit Java Studio Mobility kompilieren

Um *BlueDating* zu kompilieren, muss man im Sun Java Mobility Studio den Source Code laden, den richtigen Emulator wählen und es dann kompilieren. In Sun Java Mobility Studio kann man nun den gewünschten Emulator auswählen und die Appliaktion starten. Leider bekommt man so noch nicht das gewünschte verpackte `*.jar`

Format. Dazu muss man man unter `File/New/Jar_Archives/Jar_Recipe` ein neues Jar Archiv erstellen, dem man alle nötigen Dateien mittels `Rechtsklick/Properties` beifügen muss. Ebenfalls muss man dort den Namen und das Icon des Programms angeben. Danach kann man mittels `Rechtsklick/Update_Jar` ein Jar-Archiv erstellen. Im der mitgenerierten Jad-Datei stehen zusätzliche Informationen, die ebenfalls gebraucht werden.

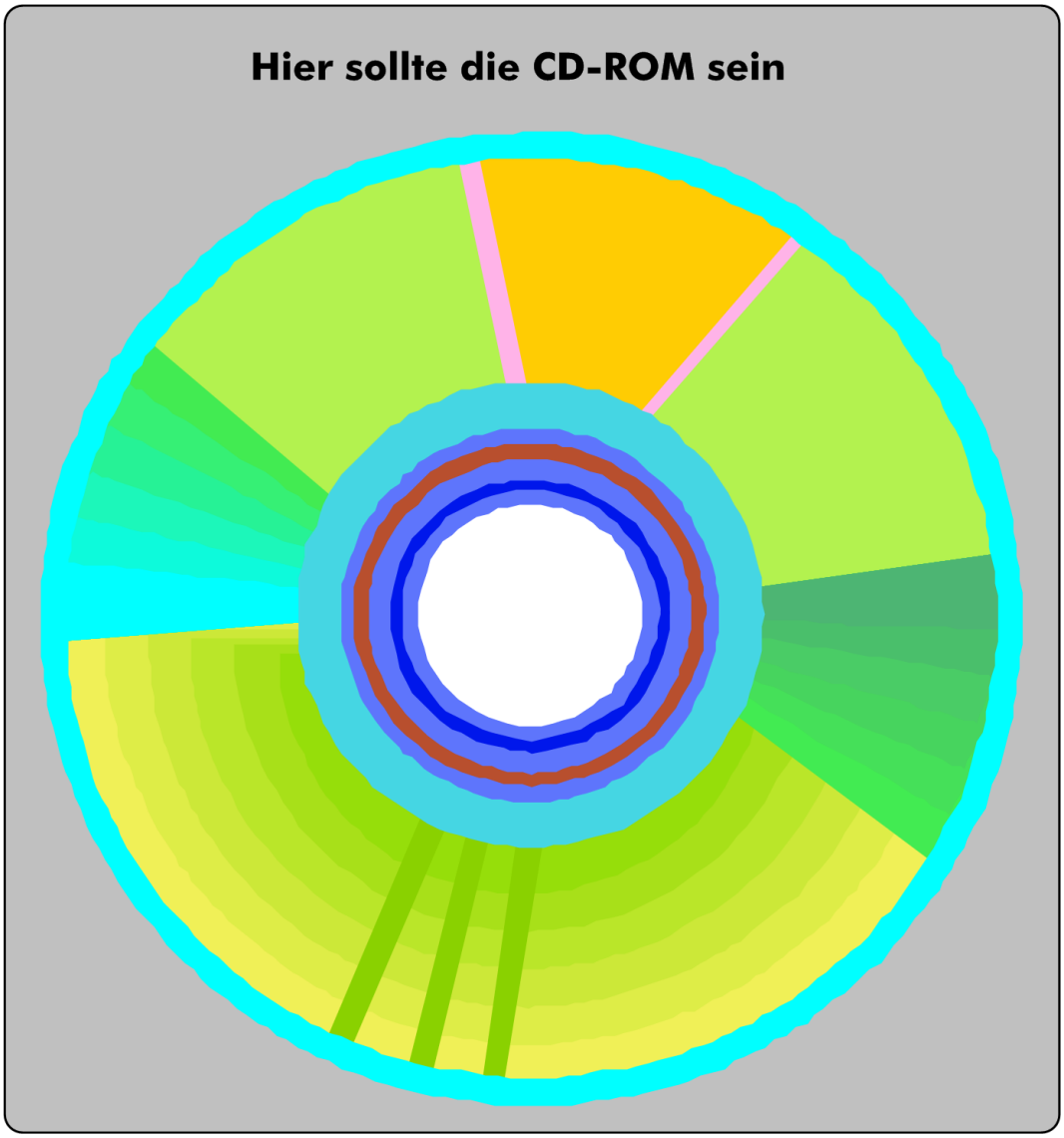
B.4.2.3 Transfer auf das Nokia 6630

Man muss den "Nokia Application Installer" der Nokia PC-Suite starten und das Nokia 6630 ans mitgelieferte USB Kabel hängen. Man benötigt nun eine kompilierte Version (ein `BlueDating.jar` und `BlueDating.jad`) und kann diese hinüberladen. B.4.2.2 beschreibt, wie man *BlueDating* kompiliert. *BlueDating* ist nun auf dem Nokia 6630 installiert und ausführbar.

Anhang C

CD-ROM

C.1 Die CD-ROM



C.2 Inhalt der CD-ROM

Verzeichnis	Inhalt
dokumentation\schlussbericht\	Enhält die PDF Schlussdokumetation
dokumentation\schlussbericht\latex\	Enhält die Schlussdokumetation in LaTeX
dokumentation\schlusspraesentation\	Enhält die Schlusspräsentation in Powerpoint
dokumentation\zwischenpraesentation\	Enhält die Zwischenpräsentation in Powerpoint
dokumente\	Enhält die Dokumente, welche frei auf dem Netz verfügbar sind
doxygen\	Enhält die Source-Code Dokumentation in PDF
doxygen\html\	Enhält die Source-Code Dokumentation in HTML
doxygen\latex\	Enhält die Source-Code Dokumentation in LaTeX
programm\binaries\	Enhält die kompilierte Version von BlueDating (JAD- & JAR-Datei)
programm\source_code\	Enhält den Source-Code von BlueDating
programm\BlueDate\	Enhält die Verzeichnisstruktur für Ktoolbar um direkt zu kompilieren

Anhang D

Aufgabenstellung

Die nächste Doppelseite zeigt die Aufgabenstellung unserer Semesterarbeit, wie sie uns zu Beginn überreicht wurde.

Institut für
Technische Informatik und
Kommunikationsnetze

Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich
Ecole polytechnique fédérale de Zurich
Politecnico federale di Zurigo

Communications Systems Research Group

Winter 2004/05

Semester Thesis

for

Sandro Schifferle (D-ITET), Christian Braun (D-ITET)

Main Advisor: Vincent Lenders
Alternate Advisors: Matthias Bossardt, Martin May

Issue Date: 18th October 2004
Submission Date: 7th February 2005

BlueDating - Bluetooth Application for Bluetooth Enabled Mobile Phones

1 Summary

Mobile phones of the latest generation feature a Java virtual machine (J2ME) and Bluetooth for short range communications. In this project, a dating application must be implemented in Java. This application allows users to enter their personal profile. Profile information is exchanged over Bluetooth as users get in communication range when moving. The mobile phone notifies a user as soon as a person is discovered whose profile matches the user's interest. This work is part of the Blue-* project [1] which aims at developing networking applications for Bluetooth-based ad-hoc networks.

2 Assignment

The goal of this thesis is to develop a working prototype of the BlueDating application.

2.1 Tasks

- Identify related work. Look for other dating applications using Bluetooth for communication.
- Familiarize yourself with Bluetooth and J2ME, specially how to access the Bluetooth stack using the JAVA API.
- Propose different technical approaches for the dating application and compare them with each other.

- Identify mobile phones with Bluetooth support which are suitable for the BlueDating application. Categorize different models based on a set of chosen characteristics. This task should be done in collaboration with the participants of the "Bluetella" thesis.
- Define the format for user profiles/interests and a mechanisms to match those.
- Design the required protocols to exchange information between mobile phones over Bluetooth.
- Implement the application.
- Test the application. Use as many devices as possible in order to detect unforeseen problems.
- Set up a demonstrator with mobile phones that shows how the application works.

2.2 Deliverables

- At the end of the second week, a detailed time schedule of the semester thesis must be given and discussed with the advisor.
- At half time of the semester thesis, a short discussion of 15 minutes with the professor and the advisor will take place. The student has to talk about the major aspects of the ongoing work. At this point, the student should already have a preliminary version of the written report, including a table of contents. This preliminary version should be brought along to the short discussion.
- At the end of the semester thesis, a presentation of 20 minutes must be given during the TIK or the communication systems group meeting. It should give an overview as well as the most important details of the work and the demonstrator should be presented at this time.
- The final report may be written in English or German. It must contain a summary written in both English and German, the assignment and the time schedule. Its structure should include an introduction, an analysis of related work, and a complete documentation of all used software tools. Three copies of the final report must be delivered to TIK.

References

[1] *The BlueStar Project*, http://www.csg.ethz.ch/research/running/Blue_star, October 2004.

18th October 2004

Prof. B. Plattner

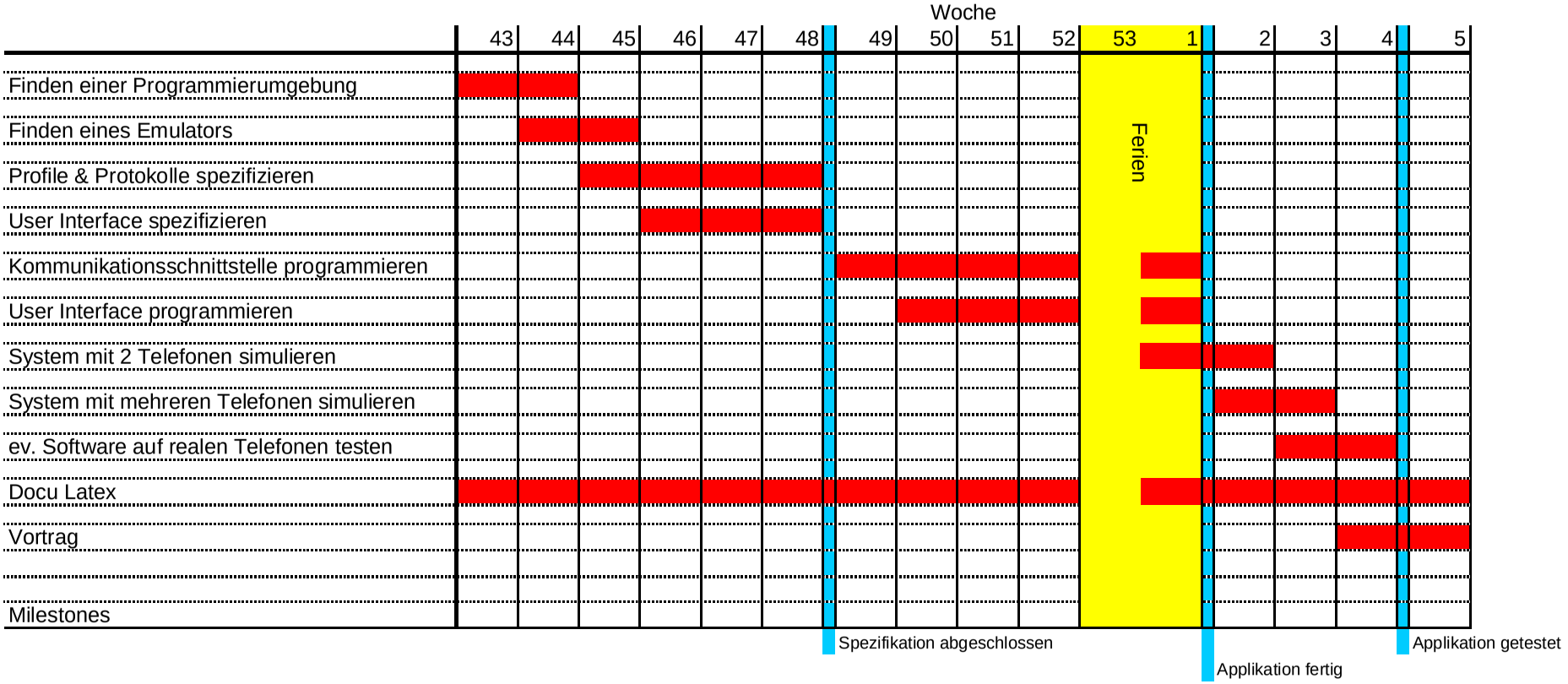
Anhang E

Zeitlicher Ablaufplan

Auf der nächsten Seite ist der zeitliche Ablauf, welchen wir zu Beginn unserer Semesterarbeit ausgearbeitet haben, abgedruckt.

Zeitplan Semesterarbeit: BlueDating - Bluetooth Application for Bluetooth Enabled Mobile Phones

Schifferle Sandro & Braun Christian



Anhang F

Abkürzungsverzeichnis

Abkürzung	Bedeutung
API	Application Program Interface
BNEP	Bluetooth Network Encapsulation Protocol
CDC	Connected Device Configuration
CLDC	Connected Limited Device Configuration
CPU	Central Processing Unit
FP	Foundation Profile
GUI	Graphical User Interface
HCI	Host Controller Interface
HID	Human Interface Device Protocol
HTML	Hypertext Markup Language
HTTP	Hypertext Transfer Protocol
J2EE	Java 2 Enterprise Edition
J2ME	Java 2 Micro Edition
J2Se	Java 2 Standard Edition
JAD	Java Application Descriptor
JAR	Java Archiv
JSR	Java Specification Request
JVM	Java Virtual Machine
KVM	Kilobyte Virtual Machine
L2CAP	Logical Link Control and Adaptation Protocol
LMP	Link Manager Protocol
MIDlet	MIDP-Applet
MIDP	Mobile Information Device Profile
MTU	Maximum Transfer Unit
OBEX	Object Exchange
PDAP	Personal Digital Assistant Profile
RMS	Record Management System
SDK	Software Development Kit
SDP	Service Discovery Protocol
TCS	Telephony Control Protocol
URL	Uniform Resource Locator
USB	Universal Serial Bus
UUID	Universal Unique Identifier
WAP	Wireless Access Protocol

Literaturverzeichnis

[1] *The BlueStar Project*,
http://www.csg.ethz.ch/research/projects/Blue_star, Oktober 2004

[2] *Bruce Hopkins, Ranjith Antony. Bluetooth for Java*,
Apress 2003,
ISBN: 1-59059-078-3

[3] *Developing Applications with the Java APIs for Bluetooth (JSR-82)*,
http://www.microjava.com/articles/Bluetooth-jsr-82-training.pdf

[4] *Java APIs for Bluetooth Wireless Technology (JSR-82)*,
http://jcp.org/aboutJava/communityprocess/final/jsr082/index.html

[5] *Die Liste der schon vergebenen UUID Nummern*
http://www.bluetooth.org/assigned-numbers

[6] *Forum Nokia: JAVA MIDP Application Developer's Guide for Nokia Devices*,
http://www.forum.nokia.com/ndsCookieBuilder?fileParamID=2655

[7] *Forum Nokia: MIDP 2.0: Introduction*,
http://www.forum.nokia.com/ndsCookieBuilder?fileParamID=6516

[8] *Forum Nokia: Introduction to Developing Networked MIDlets using Bluetooth*,
http://www.forum.nokia.com/ndsCookieBuilder?fileParamID=5005

[9] *Merlin Tonka. Möglichkeiten und Grenzen von J2ME*,
http://www2.informatik.uni-erlangen.de/Lehre/SA-DA/old-
/SA-abgeschlossen/tonka.pdf

[10] *Builder.com - Exploring J2ME: Using the Record Management System*,
http://builder.com.com/5100-6370-1050631.html

[11] *FileConnection Optional Package 1.0 Specification*,
http://jcp.org/aboutJava/communityprocess/final/jsr075/index.html

[12] *Nokia 6630 Funktionalitäten*,
http://www.nokia.com/nokia/0,8764,58711,00.html

[13] *Forum Nokia: Technical Library v1.2*,
http://www.forum.nokia.com/main/0,6566,1_48,00.html

[14] *Social Serendipity: Proximity Sensing and Cueing*,
MIT Media Laboratory Technical Note 580, Mai 2004

[15] *Die entstehung von Partnerbeziehung online*,
http://www.suz.unizh.ch/partnerwinner

[16] *Institut für Technische Informatik und Kommunikationsnetze der ETH Zürich*,
http://www.tik.ee.ethz.ch

[17] *Departement Informationstechnologie und Elektrotechnik der ETH Zürich*,
http://www.ee.ethz.ch

[18] *ETH Zürich*,
http://www.ethz.ch

[19] *swissflirt Webseite*,
http://www.swissflirt.ch

[20] *flirt & date Webseite*,
http://www.flirt.ch

[21] *PartnerWinner.ch Webseite*,
http://www.partnerwinner.ch

[22] *Serendipity Webseite*,
http://www.mobule.net

[23] *phunkz Webseite*,
http://phunknetz.de

[24] *Spotme Webseite*,
http://www.spotme.ch

[25] *nTAG Webseite*,
http://www.ntag.com

[26] *BEDD Webseite*,
http://www.bedd.com

[27] *Smart Dater Webseite*,
http://www.pweb.de/khaleque.f

[28] *Proxidating Webseite*,
http://www.proxidating.com/

[29] *CrowdSurfer Webseite*,
http://www.smallplanet.net

[30] *smile dating Webseite*,
http://www.smiledating.co.uk

[31] *Mobiluck Webseite*,
http://www.mobiluck.com

[32] *BuZZone Webseite*,
http://www.buzzone.net/

[33] *Speck Webseite*,
http://speck.randomfoo.net