Fachbereich 4: Informatik

Spontaneous (WLAN) Guest Access

- SpoGA -

S t u d i e n a r b e i t

im Studiengang Informatik

vorgelegt von

Matthias Ehrenstein

Betreuer: Dipl.-Inform. Stefan Stein
Institut für Wirtschafts- und Verwaltungsinformatik,
FB4: Informatik

Erstgutachter: Prof. Dr. J.Felix Hampe
Institut für Wirtschafts- und Verwaltungsinformatik,
FB4: Informatik

Koblenz, im Juni 2007

Vorwort

Diese Studienarbeit beschreibt den 1. Teil eines Gemeinschafts-Projektes, das mit insgesamt 3 Personen unter dem Namen „SpoGA + E-IMS“ an der Universität Koblenz-Landau, Campus Koblenz im Jahre 2006 durchgeführt wurde.

Folgende Personen wirkten an diesem Projekt mit:

- Matthias Ehrenstein (Teilprojekt SpoGA-Server)
- Christoph Speich (Teilprojekt E-IMS-Server/Groupware)
- Markus Müller (Teilprojekt E-IMS-Server/Groupware)

Alle Personen sind Studenten der Informatik, Schwerpunkt Wirtschaftsinformatik im 7. Semester an der Universität Koblenz-Landau, Campus Koblenz. Eine detaillierte Beschreibung der Themengebiete ist der Einleitung dieser Arbeit zu entnehmen.

Die Betreuung erfolgte durch Dipl.-Inform. Stefan Stein und die Begutachtung durch Prof. Dr. J.Felix Hampe, beide aus dem Institut für Wirtschafts- und Verwaltungsinformatik der Universität Koblenz-Landau, Campus Koblenz.

Als räumliche Testumgebung diente das Labor von Prof. Dr. Hampe, wo die Server installiert und in das Netzwerk der Universität integriert wurden.

Danksagung

Ich möchte mich an dieser Stelle bei allen Personen bedanken, die mich bei der Erstellung dieser Arbeit unterstützt haben. Ganz besonders zu erwähnen ist an dieser Stelle mein Betreuer Herr Dipl. Inform. Stefan Stein.
Erst durch ihn wurde ich auf dieses interessante Themengebiet aufmerksam. Zudem stand er mir in kritischen Momenten stets mit Rat und Tat zur Seite. Nochmals, vielen Dank Herr Stein.

Weiterhin danke ich meinen Korrektur-Leserinnen und -Lesern für ihre geduldige Hilfe beim Auffinden von Fehlern, sowie für die interessanten und hilfreichen Diskussionen. Zuguterletzt bedanke ich mich bei Markus Müller und Christoph Speich für die gute und angenehme Zusammenarbeit während der Dauer des Gesamt-Projektes.

Eidesstattliche Erklärung

Ich versichere, dass ich die vorliegende Arbeit selbstständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel benutzt habe. Diese Arbeit hat in gleicher oder ähnlicher Form noch keiner anderen Prüfungsbehörde vorgelegen oder wurde von dieser als Teil einer Prüfungsleistung angenommen. Alle Ausführungen, die wörtlich oder sinngemäß übernommen wurden, sind als solche gekennzeichnet.

Die Richtlinien der Arbeitsgruppe für Studien- und Diplomarbeiten habe ich gelesen und anerkannt, insbesondere die Regelung des Nutzungsrechts.

Mit der Einstellung dieser Arbeit in die Bibliothek bin ich einverstanden.

ja ☐ nein ☐

Der Veröffentlichung dieser Arbeit im Internet stimme ich zu.

ja ☐ nein ☐

Koblenz, _____ _____

Unterschrift

Inhaltsverzeichnis

1 Einleitung

Das Internet bietet im privaten sowie gewerblichen Umfeld Zugang zu Informationen aus allen erdenklichen Bereichen des Alltags, vor allem die Möglichkeit der Kommunikation und Recherche. Gerade im gewerblichen Bereich sind viele Prozesse auf diese Technologie angewiesen und Unternehmen stellen ihren Mitarbeitern die dazu nötige Infrastruktur zur Verfügung. Außerhalb des Firmengeländes besteht hingegen oft nur die Möglichkeit, das Internet über schmalbandige und kostenintensive Mobilfunkverbindungen zu nutzen. Beispielsweise besucht ein Berater eine Firma. Dieser benötigt für seine Arbeit einen Internetzugang. Dabei ist die firmeneigene Infrastruktur (oft in Form eines WLAN (Wireless Local Area Network) vorhanden, kann aber aus Sicherheitsgründen oder durch einen erhöhten Arbeitsaufwand zur Freischaltung durch einen Administrator nicht genutzt werden.

Das Problem, diese bereits vorhandene Infrastruktur für firmenfremde Gäste zugänglich zu machen, dabei die Sicherheit des Firmen-Netzwerkes zu garantieren und den Administrator zu entlasten, wurde prototypisch im Rahmen dieses Projektes SpoGA (Spontaneous WLAN Guest Access) realisiert. Hierbei erhalten ausgesuchte Firmenmitarbeiter eine Gastgeber-Berechtigung und können Gästen (wie z.B. Beratern) über eine Weboberfläche einen zeitlich begrenzt Zugang zum WLAN einrichten, ohne dabei besondere Kenntnis der Technik haben zu müssen. Die Administratoren der Firma sind somit entlastet und jeder Gastgeber ist für seine Gäste verantwortlich.

Eine solche manuelle Einrichtung für Gast-Zugänge ist für eine kleine Anzahl von Gästen praktikabel. Im Falle von Konferenzen oder größeren Veranstaltungen aber zu zeitaufwendig für die Gastgeber. Um eine Automatisierung der Zugänge auch für solche Umstände zu gewährleisten, wurde aufbauend auf SpoGA ein E-IMS-System (Extended Invitation Management Server) realisiert. Dieses ermöglicht dem Gastgeber Veranstaltungen anzulegen und zu verwalten, sowie die dazu notwendigen Ressourcen (z.B. Räume oder Beamer) zu buchen, falls der Gast selbst als Vortragender fungiert. E-IMS übernimmt dabei das Einladen der Gäste, wertet deren Zu- bzw. Absagen aus und erstellt über eine Webservice-Schnittstelle mit dem SpoGA-System deren Zugänge. Wird dieses E-IMS-System nur selten benötigt, so kann es durch einen Dienstleister angeboten werden, anderenfalls auch in die Firma als eigener Server integriert werden.

Während der Durchführung des Projektes haben sich mehrere Fragen ergeben. Die zentralen Aspekte sind dabei kurz festzuhalten, und sollen im Laufe dieser Arbeit beantwortet werden:

- **Wie realisiert man zeitnahe Gastzugänge?**
- **Wie kann man in diesem Fall den Administrator entlasten?**
- **Wird dadurch die Sicherheit des Firmennetzes gefährdet?**
- **Inwieweit muss die vorhandene Firmen-Infrastruktur ergänzt oder verändert werden um einen solchen Ansatz zu realisieren?**
- **Gibt es durch den Einsatz dieses Systems einen finanziellen Mehraufwand oder können sogar Kosten eingespart werden?**

Das folgende Kapitel beschreibt zunächst einen groben Überblick über den Umfang des Gesamt-Projektes SpoGA+E-IMS, welches in 3 Studienarbeiten[1] realisiert wurde. Dabei werden kurz die wichtigsten Aspekte und Zusammenhänge der beiden Systeme erläutert. Anschließend wird in einer Anforderungsanalyse der Ist- und Sollzustand eines Beispielszenarios ermittelt, um die dabei vorhandenen Probleme sowie mögliche Lösungsansätze darzustellen und die Relevanz des SpoGA-Systems zu verdeutlichen. In den darauffolgenden Kapiteln soll die technische Umsetzung des Systems bis hin zur Realisierung beschrieben werden und abschließend erfolgt eine Zusammenfassung der erreichten Ziele.

[1] Das Gesamt-Projekt SpoGA und E-IMS besteht aus dieser Studienarbeit, der Studienarbeit von Markus Müller [Mül07] und der Studienarbeit von Christoph Speich [Spe07]

2 Aufbau des Projektes

2.1 Beschreibung und Aufteilung der Arbeiten

Das Gesamtprojekt wurde in Form von 3 Studienarbeiten realisiert:

1. **Spontaneous (WLAN) Guest Access (SpoGA)**

   Es soll ein System entwickelt werden, welches auf der bestehenden Netzwerkinfrastruktur einer beliebigen Firma aufsetzt und temporäre WLAN-Zugänge und somit Zugang zum Internet/Firmennetz ermöglicht (Spontaneous (WLAN) Guest Access, kurz SpoGA). Ein solcher (Gast-)Zugang soll eigenständig durch einen Gastgeber (Mitarbeiter dieser Firma) eingerichtet werden können, ohne dabei einen Administrator mit einzubeziehen.

   Das SpoGA-System fungiert hierbei als Router zwischen einem vorhandenen Firmen-WLAN und dem Internet. Gästen wie z.B. Beratern soll somit die Möglichkeit gegeben werden, schnell und unkompliziert einen Internetzugang, den sie zur Ausübung ihrer Arbeit benötigen, nutzen zu können. Dabei soll die Sicherheit des Firmennetzes nicht gefährdert sein, und die Administratoren entlastet werden. Der SpoGA-Server soll sowohl über eine Webseite als auch über eine Webservice-Schnittstelle erreichbar sein. Ein Gast soll einen persönlichen Code erhalten, um sich mit diesem am System anzumelden und somit Zugang zum Internet zu erhalten.

2. **Extended Invitation Management System (E-IMS)-Modul Veranstaltungsverwaltung ([Mül07])**

   E-IMS (Extended Invitation Management Server) ist ein optionales System, welches von einem zentralen Dienstleister angeboten werden kann, oder bei häufiger Nutzung auch in die Netz-Infrastruktur der Firma selbst integrierbar ist. Es dient als Erweiterung von SpoGA und basiert auf einer Groupware zur Planung und Verwaltung von Veranstaltungen. Als potenzielle Nutzer werden hier Firmen betrachtet, die größere Veranstaltungen wie Fortbildungen, Präsentationen oder firmenübergreifende Konferenzen ausrichten. Es soll den Nutzern die Möglichkeit bieten, ihre Veranstaltungen zu planen und zu organisieren. Angefangen bei der Terminplanung, über das Versenden der Einladungen und der Reservierung von Örtlichkeiten sowie Ausstattung der Konferenzräume. Zusätzlich lassen sich über dieses System die Gastzugänge eines SpoGA-Servers verwalten und es besteht die Möglichkeit der Kostenabrechnung.

   Die erste Ausarbeitung des E-IMS-Systems behandelt das Verwalten von Firmen- und Benutzerdaten, sowie das Erstellen von Veranstaltungen am E-IMS-System. In der Groupware des E-IMS soll hierzu eine Funktion erstellt werden, welche neue Firmengruppen anlegt, denen dann Benutzer hinzugefügt werden. Danach sollen über ein in der Groupware geschaffenes Modul neue Veranstaltungen erstellt und verwaltet werden. Der Gast kann Räume oder Gerätschaften buchen und wird von

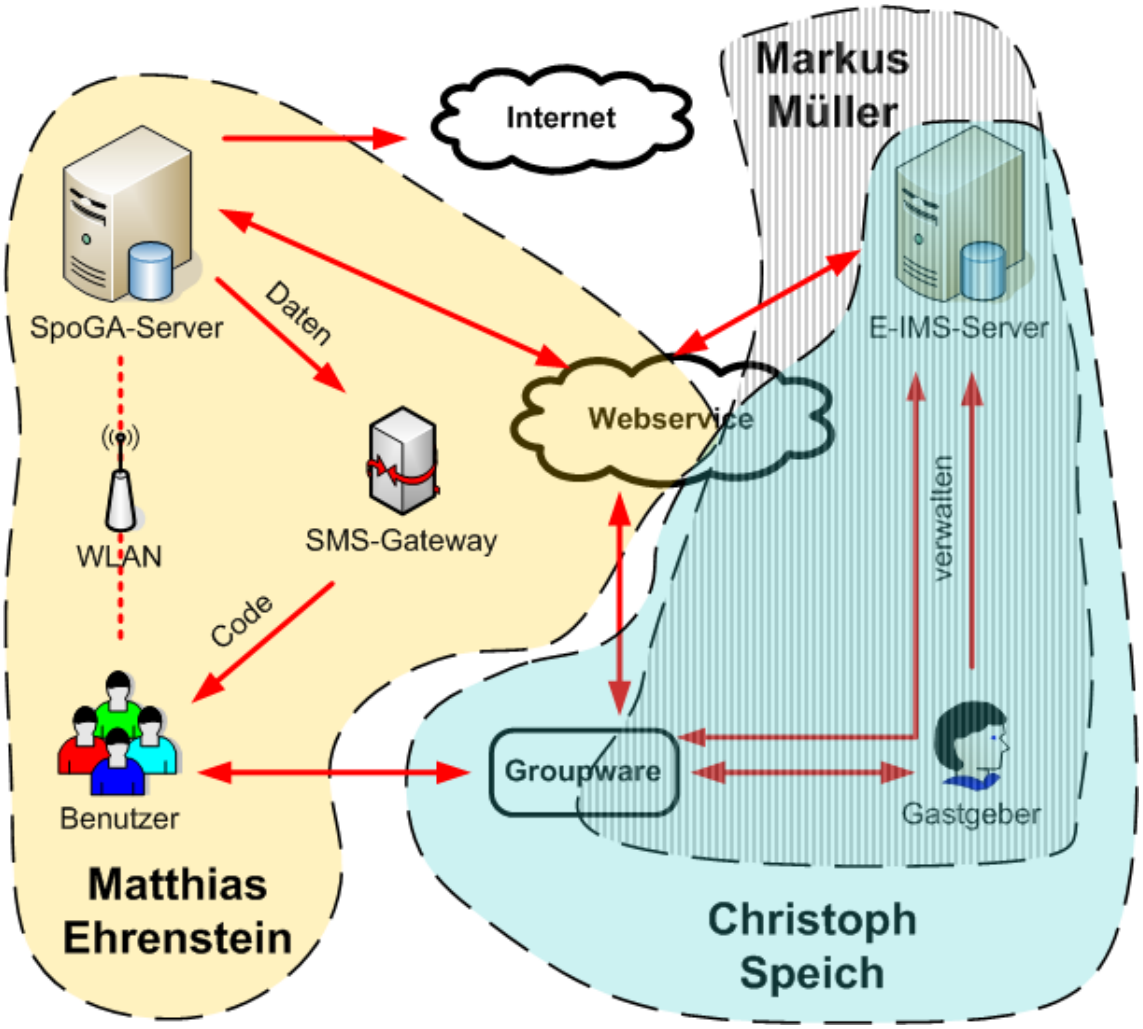


Abbildung 2.1: Übersicht Gesamtprojekt

einem Gastgeber eingeladen. Als weiteres Modul soll im Rahmen dieser Arbeit ein Abrechnungsmodul erstellt werden. Hierüber können den einzelnen Gästen einer Veranstaltung im Anschluss eine Rechnung für die in Anspruch genommenen Leistungen gestellt werden.

3. **Extended Invitation Management System (E-IMS)-**
   Modul SpoGA-Manager und Erweiterung der Ressourcenverwaltung
   ([Spe07])

   Diese Arbeit stellt die Schnittstelle zwischen den beiden anderen Arbeiten dar. Hier werden die im E-IMS-System erstellten Veranstaltungen und die daraus resultierenden Teilnehmerlisten dazu verwendet, einen am Veranstaltungsort befindlichen SpoGA-Server mit allen notwendigen Daten zu versorgen. Dazu müssen den verschiedenen Gästen der Veranstaltung durch den Gastgeber Zugänge bereitgestellt und die dafür notwendigen Anmeldecodes übermittelt werden. Bevor dies geschehen kann, müssen beide Systeme durch eine einmalige Konfiguration aufeinander abgestimmt werden. In einem weiteren Teil der Arbeit wurde die (bereits in der Groupware des E-IMS-Systems integrierte) Ressourcenverwaltung für die Kombination mit dem SpoGA-System angepasst und erweitert. Diese Arbeit bildet somit die Datengrundlage des SpoGA-Servers bei gemeinsamer Nutzung mit E-IMS und der Onlinezeit-Abrechnung.

   Die beiden Systeme sollen über eine Webservice-Schnittstelle miteinander kommunizieren. Sie sind eigenständig oder in Kombination einsetzbar. Ein E-IMS-Server kann mehrere SpoGA-System verwalten.

Die Aufteilung des Gesamtprojektes ist in Abbildung 2.1 dargestellt. Diese Arbeit beschäftigt sich mit der Entwicklung des SpoGA-System, Markus Müller und Christoph Speich befassten sich mit der Entwicklung des E-IMS-Systems ([Mül07],[Spe07]).

2.2 Themenübergreifende Arbeiten

Einige Arbeitsschritte wurden gemeinsam im Team bewältigt. Darunter fallen im Einzelnen:

- die grundlegende Planung und Strukturierung des Gesamt-Projektes SpoGA und E-IMS
- der Entwurf einer Architektur des Gesamt-Projektes
- das Testen und Bewerten bestehender Groupware-Systeme als Basis des E-IMS-Systems
- ein Interview mit dem wissenschaftlichen Leiter des Rechenzentrums der Universität Koblenz zum Thema Groupware
- Strukturierung und Planung der verwendeten Datenbanken sowie der Webservice-Schnittstelle

3 Anforderungsanalyse

In diesem Kapitel werden die Erhebungsmethoden und die Ergebnisse der Anforderungsanalyse beschrieben. Zunächst folgt ein Überblick über die theoretischen Grundlagen, wie z.B. die Methoden zur Erhebung einer solchen Analyse. Die weiteren Abschnitte beschreiben die die Anforderungen und Aufgaben des zu entwickelnden Prototypen.

Die Anforderungsanalyse teilt sich in zwei Schritte, die Ist-Analyse und die Soll-Analyse. In Kapitel 3.2.1 wird der Ist-Zustand einer WLAN-Registrierungsprozedur für Gäste am Beispiel eines möglichen Szenarios dargestellt. Dabei ist zu beachten, dass die Schwachstellen der aktuellen Lösung aufgezeigt werden. Durch gezielte Erhebung von Informationen sollen diese Schwachstellen identifiziert werden. Ein weiterer Aspekt bei der Erhebung des Ist-Zustandes ist, die Stärken und Vorteile der Ist-Lösung aufzuzeigen, also alles was sich bisher bewährt hat und möglichst unverändert bleiben soll, um diese dann in der Soll-Lösung wieder zu integrieren.

Anschließend wird in Kapitel 3.2.2 ein Zustand beschreiben, wie er nach der Implementierung des SpoGA-Systems aussehen könnte (Soll-Zustand). Hier werden die inhaltlichen und ablaufbezogenen Anforderungen an das zu entwickelnde System beschrieben. Einerseits ergeben sich diese automatisch aus den Nachteilen des Ist-Zustandes, andererseits auch durch die Erhebung weiterer Informationen.

3.1 Methoden der Erhebung

Für die allgemeine Erhebung der benötigten Informationen stehen mehrere Methoden zur Verfügung. Zum einen können die Informationen bei unterschiedlichen Personengruppen ermittelt werden, zum anderen durch unterschiedliche Erhebungstechniken. In diesem Abschnitt werden kurz die unterschiedlichen Personengruppen identifiziert, sowie eine Auswahl von möglichen Erhebungstechniken aufgelistet.

3.1.1 Personengruppen bestimmen

Zu Beginn der Anforderungsanalyse sollten die Personengruppen bestimmt werden, die für die Erhebung des Ist- und Soll-Zustandes in Frage kommen, um mögliche Schwachstellen oder Verbesserungsvorschläge zu erkennen. Dabei beschränkt man sich auf Personen, die unmittelbar mit dem vorhandenen System arbeiten, oder das zu entwickelnde System nutzen werden. Dies sind zum einen die direkten Anwender (Gastgeber, Gäste, Administrator), zum anderen die indirekten Anwender wie z.B. Controller oder Buchhalter, die die entstandenen Daten (z.B. zur Kostenabrechnung der Systemnutzung) weiterverarbeiten. Letztere müssen bei der Anforderungsanalyse des SpoGA-Systems nicht berücksichtigt werden, da die Kostenabrechnung und Auswertung nicht in dessen Aufgabenbereich liegt.

3.1.2 Erhebungstechniken

Für die Beschaffung von Information können verschiedene Erhebungstechniken angewandt werden. Es können allerdings nicht für jede Aufgabenstellung alle Techniken eingesetzt werden. Daher wird in diesem Abschnitt nur eine Auflistung der für diese Arbeit in Frage kommenden Erhebungstechniken (Auswertung von vorhandenen Unterlagen, Interview, Beobachtung) vorgestellt. Die Erhebungstechniken wurden aus [PS04] entnommen.

- **Auswertung von vorhandenen Unterlagen**
  Die Auswertung von vorhandenen Unterlagen (beispielsweise Organisationspläne oder Arbeitsplatzbeschreibungen) ist vor allem geeignet, um einen Überblick über die Anforderungen an das zu entwickelnde System zu erhalten. Es können dadurch meist Aussagen über die Struktur und die Aufgabenverteilung der Nutzer gewonnen werden. Der Hauptvorteil dieser Erhebungstechnik liegt darin, dass die Informationen schon erhoben wurden und somit nicht neu gesammelt, sondern lediglich ausgewertet werden müssen. Dies bedeutet Zeit- und dadurch auch Kostenersparnis. Es ist eine der zentralen Erhebungstechniken bei der Bestimmung des Ist-Zustandes. Problematisch hierbei ist aber, dass die Unterlagen oft nicht in dem Zustand sind, der für die Analyse notwendig ist. Dies kann z.B. der Fall sein, wenn der Zweck der Erstellung ein anderer, das Alter der Unterlagen zu hoch oder der Detailierungsgrad zu grob oder zu fein ist. Ein weiterer nicht zu vernachlässigender Gefahrenpunkt ist die Möglichkeit der Manipulation der Informationen durch einen subjektiven Erheber.

- **Interview**
  Ein Interview ist die gezielte Befragung von Personen, die Hintergrundwissen über das Themengebiet besitzen. Es kann je nach Interviewpartner in einer unterschiedlichen Art der Strukturierung ablaufen. Die Art des Interviews reicht von einem einfachen Gespräch, welches ohne Strukturierung geführt wird bis zu standardisierten Interviews mit geschlossenen Fragen, bei denen der Befragte eine Auswahl von vorgegebenen Antworten hat. Der Vorteil des Interviews besteht darin, dass bei jeder Person unterschiedlich stark auf die einzelnen Fragen eingegangen werden kann. Erscheint eine Frage für den Befragten wichtig, dann kann in dieser Richtung weitergefragt und dadurch weitere Information ermittelt werden. Das Interview kann für alle Personengruppen von Vorgesetzten (Gespräch) bis Sachbearbeiter (Standardisiertes Interview) angewendet werden. Durch die direkte Rückmeldung des Befragten kann eine unklare Frage anders formuliert und erneut gestellt werden. Nachteil des Interviews ist der hohe Zeitbedarf der Befragung und Auswertung. Vor allem bei Gesprächen ist die Auswertung und Extraktion der Information schwierig und zeitaufwendig.

- **Beobachtung**
  Bei der Beobachtung wird einem Beobachteten eine Person (Beobachter) zugeordnet, die seine Arbeitsschritte aufzeichnet. Der Vorteil besteht darin, dass die Kontrolle dieser Aufzeichnung gewährleistet ist. Damit werden die relevanten Arbeitsschritte erfasst, eine unbewusste Manipulation durch Dritte wird so ausgeschlossen. Der Beobachter kann die Information in der Dichte erfassen, die er für notwendig hält. Besonders gut geeignet ist diese Technik, wenn nicht der komplette Arbeitsablauf

ermittelt werden soll, sondern nur ein Teilbereich. Nachteilig ist zunächst für den Analytiker der hohe Zeitbedarf der Beobachtung. Viele der beobachteten Tätigkeiten wiederholen sich oft, bis wieder neue Tätigkeiten auftreten. Die Technik eignet sich daher nicht für längere Routinebearbeitungen, bei denen nur selten Sonderfälle auftreten, die beobachtet werden sollen.

- **Weitere Erhebungstechniken**
  Außer den aufgeführten Erhebungstechniken gibt es natürlich noch eine große Anzahl weiterer Techniken, die allerdings für dieses Projekt nicht in Frage gekommen sind. Darunter fallen z.B. Fragebogen, Selbstaufschreibung oder Gruppengespräche.

Dies sind die 3 wesentlichen Techniken, die für die Erhebung des Ist-Zustandes des SpoGA-Projektes angewandt wurden. Als zu befragenden Personengruppe dienten Mitarbeiter einer kleinen Firma. Als Beobachtungs-Personen haben sich dessen Mitarbeiter zur Verfügung gestellt. Interview-Partner zur Auswahl der Groupware für das E-IMS-System (siehe Anhang B) war Uwe Arndt vom Rechenzentrum der Universität Koblenz-Landau (GHR-KO)[1]. Die Ergebnisse der Analyse werden in den nächsten Abschnitten beschrieben.

3.2 Ergebnisse der Anforderungsanalyse

3.2.1 Ist-Zustand

Um die Personengruppen für die Erhebung des Ist-Zustandes zu definieren soll zunächst ein mögliches **Berater-Szenario** beschrieben werden:

Eine Firma plant eine größere Investition und läd dazu einige Finanzberater zu einem Tagungs-Gespräch mit der Geschäftsleitung ein. Jeder Berater besitzt einen WLAN-fähigen Laptop und soll Zugang zum Internet erhalten, um dadurch seine Tätigkeit ausüben zu können. Für einen solchen Zugang benötigt ein Berater eine Registrierung seiner WLAN-Karte und dessen Freischaltung durch einen Administrator dieser Firma, da der Zugriff auf das Firmen-WLAN aus Sicherheitsgründen nur registrierten Personen gestattet ist. Dies ist mit Zeit- und Verwaltungsaufwand verbunden und stellt eine potenzielle Belastung aller beteiligten Personen dar.

Von folgenden Personengruppen, bzw. Repräsentanten dieser Gruppen, können Informationen zur Anforderungsanalyse beschafft werden:

- **Gastgeber**
  Ein Gastgeber läd Gäste zu geplanten Veranstaltungen ein und muss im Vorfeld deren Zugänge von einem Administrator oder Mitarbeiter des Rechenzentrums einrichten lassen. Er kann selbst keine Zugänge erstellen oder löschen. Ein Gastgeber ist meist selbst ein Mitarbeiter des gastgebenden Unternehmens.

- **Gast**
  Ein Gast nimmt an Tagungen/Veranstaltung teil, sofern er von einem Gastgeber im Vorfeld dazu eingeladen wurde. Er braucht dazu kein technisches Wissen und will

[1] Hochschulrechenzentrum Uni Koblenz: http://www.uni-koblenz.de/GHRKO/

mit möglichst geringem Zeit- und Arbeitsaufwand Zugang zum Firmen-WLAN bzw. Internet erlangen.

- **Administrator**
  Der Administrator des Unternehmens ist berechtigt, Gastzugänge anzulegen und auch wieder zu entfernen. Er steht in engem Kontakt mit dem Gastgeber, um dessen Gäste für die Nutzung des Firmen-WLANs freizuschalten. Zudem ist er zuständig für die Sicherheit und den Betrieb der Firmen-Infrastruktur.

Abbildung 3.1 soll am Beispielszenario „Beratung“ die notwendigen Schritte verdeutlichen, um als Berater einen Zugang zum Firmen-WLAN (und damit zum Internet) zu erlangen. Sie stellt somit den Ist-Zustand dar.

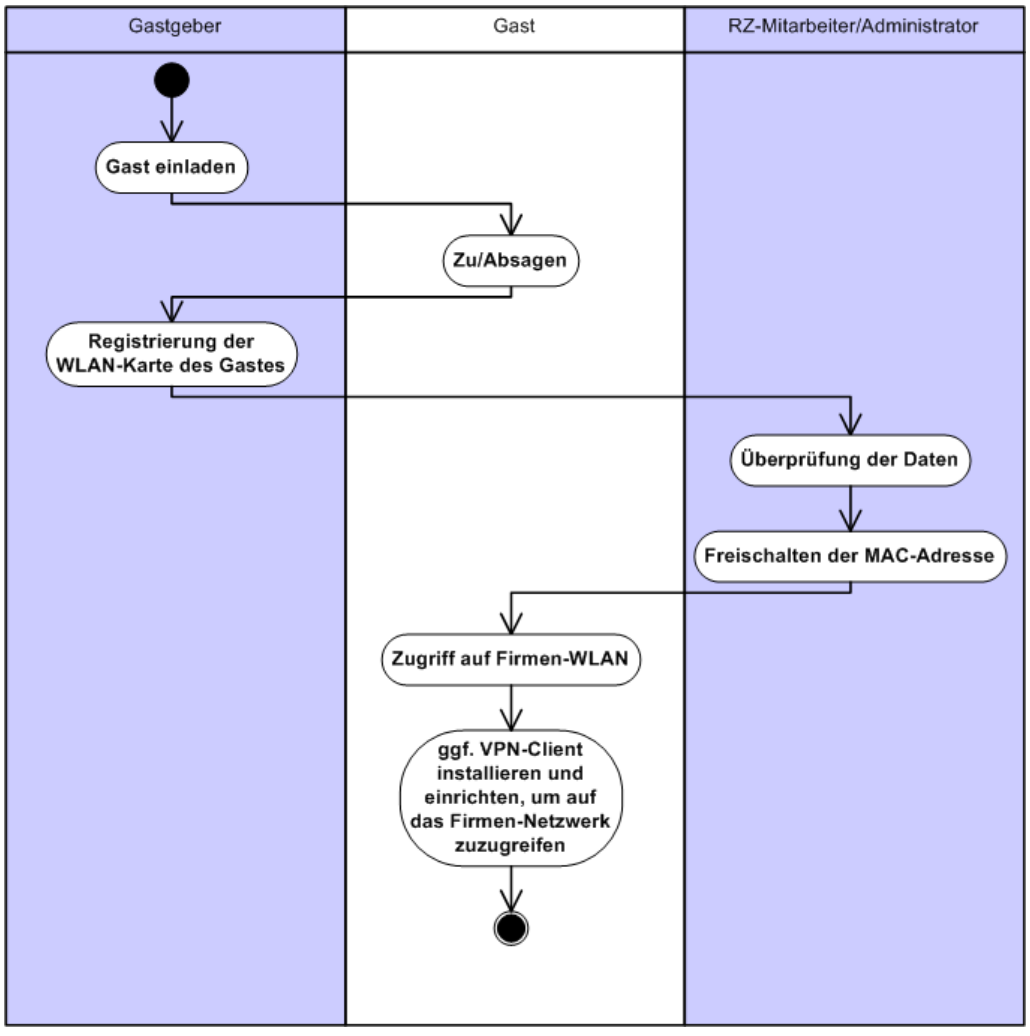


Abbildung 3.1: Ist-Zustand des Registrierungsvorgangs für WLAN-Nutzung

Zunächst benötigt ein Berater eine Registrierung seiner WLAN-Karte, um Zugang zum WLAN zu erhalten. Dies geschieht üblicherweise anhand deren MAC-Adresse, da diese separat freigeschaltet werden muss, um die Sicherheit des Firmen-WLANs vor unbefugtem Zugriff zu gewährleisten. Nach Freischaltung der MAC-Adresse durch einen Administrator oder Mitarbeiter des Rechenzentrums kann er nun das Firmen-WLAN nutzen, und somit auf das Internet zugreifen. Will ein Berater zusätzlich auf die gesicherte Firmen-Netzinfrastruktur (vom WLAN getrennt, enthält sensible Firmendaten) zugreifen, so benötig er einen VPN-Client, um eine sichere Verbindung aufzubauen.

Vorteile der Ist-Lösung:

- Zugang zum Firmen-WLAN für Gäste nicht erreichbar, erst nach manueller Freischaltung (durch einen Administrator) möglich. Somit Sicherheit vor unbefugtem Zugriff gewährleistet
- Vollständige Überwachung des Vorgangs durch einen Administrator bzw. Rechenzentrum, somit Missbrauchsversuche unwahrscheinlich

Nachteile der Ist-Lösung:

- Registrierung abhängig von Verfügbarkeit der RZ-Mitarbeiter/Administratoren (daher Registrierung nicht zeitnah möglich)
- Mögliche Sicherheitslücken durch vorhandene, nicht mehr benötigte Zugänge
- Registrierungsvorgang zur Nutzung des WLANs sehr zeitaufwendig (manuelle Freischaltung)
- Überprüfung der Nutzer-Daten durch Administrator/Rechenzentrum nötig
- Für technisch unversierte Nutzer/Gäste nicht einfach (MAC-Adresse auslesen usw.)

3.2.2 Soll-Zustand

Ziel des SpoGA-Projektes ist es, die genannten Nachteile der Ist-Lösung zu minimieren bzw. soweit möglich einzelne Schritte zu automatisieren und dabei die Vorteile beizubehalten. Dabei steht der Gast im Mittelpunkt. Ihm soll mit möglichst geringem Verwaltungsaufwand ein zeitnaher Zugang zum Firmen-WLAN/Internet bereitgestellt werden können. Am Beispiel des Berater-Szenarios würde das bedeuten, dass ein Berater keinen Kontakt mit Administratoren oder Mitarbeitern des Rechenzentrums benötigt und diese somit ebenfalls entlastet werden. Der gesamte Registrierungsvorgang soll vom SpoGA-System übernommen werden.

Dabei ändern sich die Rollen und speziell die Anforderungen der Personengruppen ebenfalls:

- **Gastgeber**
  Ein Gastgeber ist ein Benutzer des SpoGA-Systems mit besonderen Privilegien. Er hat die Erlaubnis des Administrators, seine Gäste für die Nutzung des Firmen-WLANs freischalten, oder diese wieder zu löschen und übernimmt die Verantwortung für seine Gäste. Ein Gastgeber ist meist selbst ein Mitarbeiter des gastgebenden Unternehmens und steht damit in einem engen Vertrauensverhältnis zu einem Administrator.

  Für ihn sind die Brauchbarkeit, darunter vor allem die Bedienbarkeit und die Nützlichkeit entscheidend, da er regelmäßig mit dem System arbeiten wird.

- **Gast**
  Ein Gast wird von einem Gastgeber eingeladen, an Tagungen/Veranstaltung teilzunehmen. Er besitzt kein technisches Wissen über das System und will mit möglichst geringem Zeit- und Arbeitsaufwand einen Gastzugang erlangen.

  Im Gegensatz zum Gastgeber arbeitet der Gast nicht direkt mit dem System (Admin-Oberfläche), sondern benutzt nur dessen Ergebnisse (Anmeldeseite). Die Technik des zu entwickelnden Systems ist für ihn dabei nicht wichtig.

- **Administrator**
  Der Administrator des Unternehmens ist berechtigt, Gastgeber hinzuzufügen und wieder zu entfernen. Er trägt die Verantwortung für die Funktionalität des SpoGA-Systems und hat die volle Kontrolle über alle Zugänge, um das System zu überwachen, oder im Falle eines Fehlers einschreiten zu können.

  Seine Anforderungen an das System beziehen sich in erster Linie auf die Integrierbarkeit in die vorhandene Firmen-Infrastruktur. Zusätzlich muss gewährleistet werden, dass die Sicherheit des Firmennetzes nicht gefährdet ist. Ebenso spielt auch die Bedienbarkeit des Systems, speziell der Kontrollfunktionen (Logging, Auflisten der Gäste und Gastgeber), eine wichtige Rolle für den Administrator.

Der Vorgang der Freischaltung eines Gast-Zuganges in diesem Szenario könnte dabei wie in Abbildung 3.2 dargestellt aussehen.
Dabei legt der Gastgeber über die Benutzeroberfläche des SpoGA-Servers die Zugänge seiner Gäste an. Alle weiteren Aufgaben, die zuvor vom Administrator bzw. von einem RZ-Mitarbeiter verrichtet wurden, werden nun vom SpoGA-System übernommen. Es erstellt selbstständig die Zugangscodes, die die Gäste zum Anmelden am SpoGA-System benötigen und versendet diese unmittelbar vor Beginn der Nutzung per SMS an die Gäste. Der Versand per SMS hat 2 Gründe:

1. der Gast bekommt den Zugangscode **zeitnah**, was bei der Übermittlung durch andere Dienste wie z.B. E-Mail nicht gewährleistet wäre

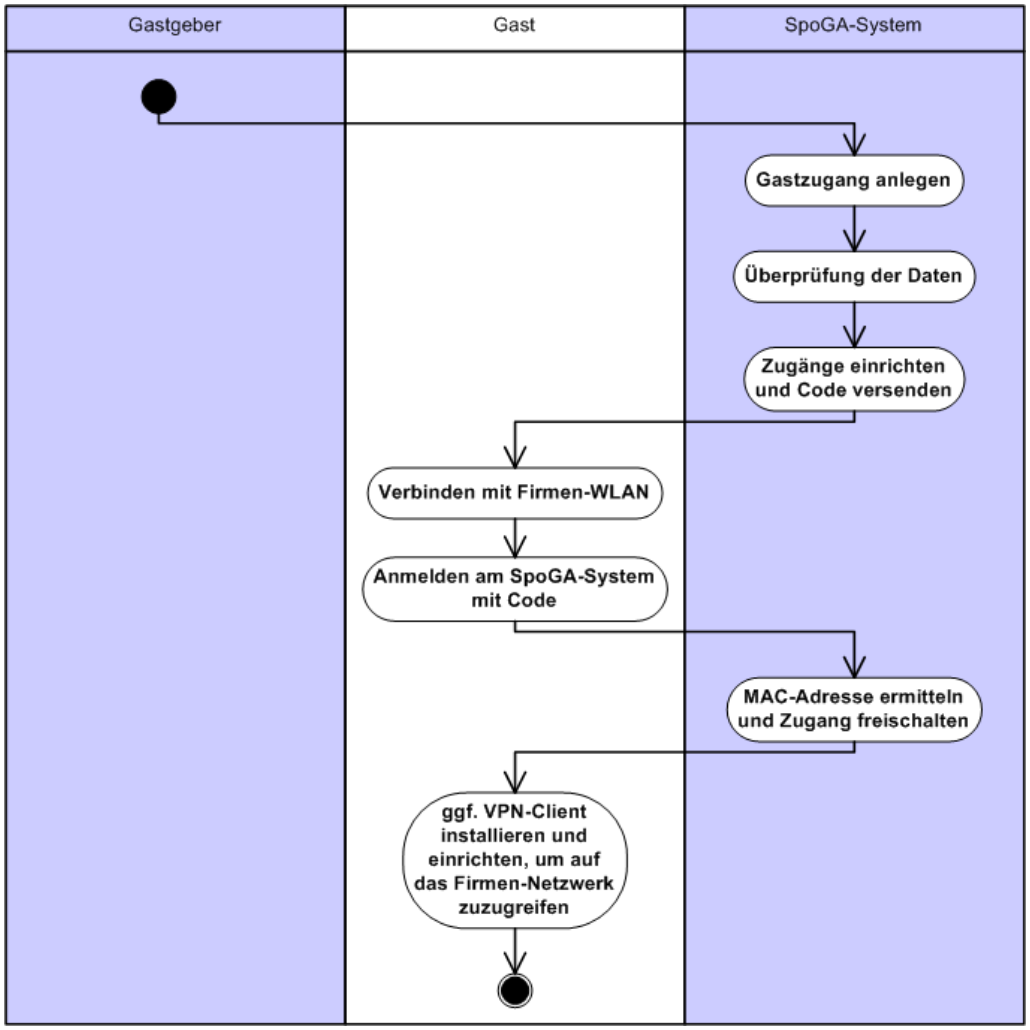


Abbildung 3.2: Registrierungsvorgang für WLAN-Nutzung mit SpoGA-System

2. der Gast erhält den Zugangscode auf sein **persönliches Endgerät** (Mobilfunkgerät), wodurch die Zugriffsmöglichkeit durch Fremdpersonen gering ist

Vergrößert man im Berater-Szenario die Anzahl der Nutzer des SpoGA-Systems und werden zusätzlich noch Gerätschaften (z.B. Beamer, Räume) für einen Vortragenden benötigt, so entsteht ein weiteres Szenario, das **Konferenz-Szenario**. Hierbei kommt zusätzlich zum SpoGA-System die Erweiterung E-IMS zum Einsatz, um eine erweiterte Benutzerverwaltung und Ressourcenplanung zu gewährleisten. Dabei kommt eine weitere Personengruppe hinzu:

- **E-IMS-Administrator (bei Nutzung des optionalen E-IMS-Systems**
  Der E-IMS-Administrator ist für die Betreuung und Konfiguration des optionalen E-IMS-Systems zuständig. Dieses kann sowohl Firmeneigentum sein, als auch out-sourced bei einem ISP (Internet Service Provider) oder sonstigem Dienstleister stehen. Im ersten Fall übernimmt der Administrator der Firma auch die Rolle des

E-IMS-Administrators.

Die Anforderungen des E-IMS-Administrators an das SpoGA-System sind dabei minimal. Die Funktionalität der Schnittstelle zwischen den beiden Systemen muss gewährleistet sein.

Im Falle des Konferenz-Szenarios mit zusätzlicher Nutzung des E-IMS-Sytems kann der Gastgeber seine Gäste bequem über die Groupware des E-IMS-Systems verwalten und einladen. Der Vorgang der Freischaltung eines Gast-Zuganges im Konferenz-Szenario könnte nach Implementierung des SpoGA-Systems (mit Nutzung des E-IMS-System) wie in Abbildung 3.3 dargestellt aussehen.

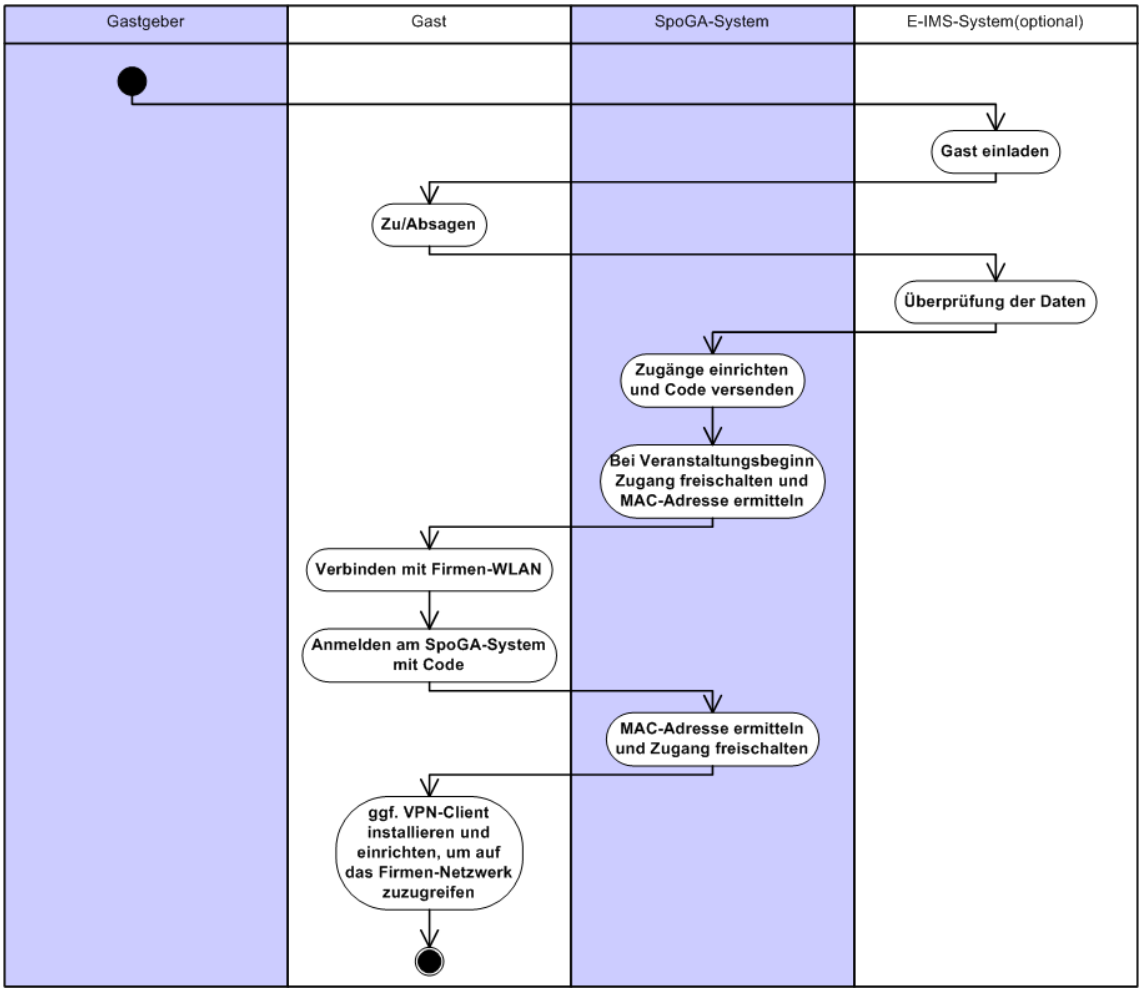


Abbildung 3.3: Registrierungsvorgang für WLAN-Nutzung mit SpoGA- und E-IMS-System

Die Einladung der Konferenz-Teilnehmer (hier Nutzer/Gast) zu einer geplanten Veranstaltung wird vom E-IMS-System übernommen. Das System sammelt daraufhin die Zu- bzw. Absagen der Teilnehmer und entscheidet welche Zugänge somit angelegt werden müssen. Diese Information teilt E-IMS dem SpoGA-System mit und SpoGA generiert nun die Gastzugänge und verschickt unmittelbar vor Veranstaltungsbeginn die Freischalt-Codes per SMS an die Teilnehmer.

Befindet sich nun ein Teilnehmer in Reichweite des Firmen-WLANs und öffnet einen Internet-Browser, so wird er auf die Benutzeroberfläche des SpoGA-Servers geleitet und kann sich dort, sobald die Veranstaltung beginnt mit seinem Zugang anmelden. Dieser Code ist nur für diesen Teilnehmer und auch nur für den Zeitraum der Veranstaltung gültig. Somit ist ein Missbrauch oder eine Weitergabe des Zuganges nicht möglich. Wird eine Verbindung zum Firmen-Netz benötigt so muss ein Teilnehmer dazu eine VPN-Verbindung nutzen, um eine sichere Kommunikation zu gewährleisten, da das Firmen-WLAN kein ausreichende Sicherheit bereitstellt (unverschlüsseltes WLAN).

Der hier geschilderte Ablauf bezieht sich auf die Nutzung von SpoGA- **und** E-IMS-System. Dabei ist das SpoGA-System auch eigenständig einsetzbar (stand-alone). Dies kann z.B. der Fall sein, wenn nur einzelne Gäste einen Zugang benötigen (Berater-Szenario) und eine manuelle Freischaltung durch den Gastgeber ausreichend ist. Hierzu wird die Administrator-Seite der SpoGA-Oberfläche eingesetzt.

Durch den Einsatz des SpoGA-Systems (optional auch in Verbindung mit E-IMS) ergeben sich daraus folgende Vorteile gegenüber der Ist-Lösung:

- Nach einmaliger Bereitstellung des Systems kein Administrator/RZ-Mitarbeiter mehr nötig, höchstens zu Kontrollzwecken
- automatisierter Vorgang zum Erstellen und Versenden der Zugangscodes für die Gäste
- einfache Bedienbarkeit auch für technisch unversierte Nutzer
- spontane Nutzung möglich (auch an Wochenenden)
- stand-alone-Einsatz des SpoGA-System möglich (E-IMS optional, nicht zwingend erforderlich)

Die Vorteile der Ist-Lösung, insbesondere die Kontrollfunktion des Administrators und die Sicherheit des WLANs vor unbefugtem Zugriff, werden dabei nicht beeinträchtigt und bleiben weiterhin bestehen. Zum besseren Verständnis ist in Abb. 3.4 die Architektur des zu entwickelnden Systems dargestellt.

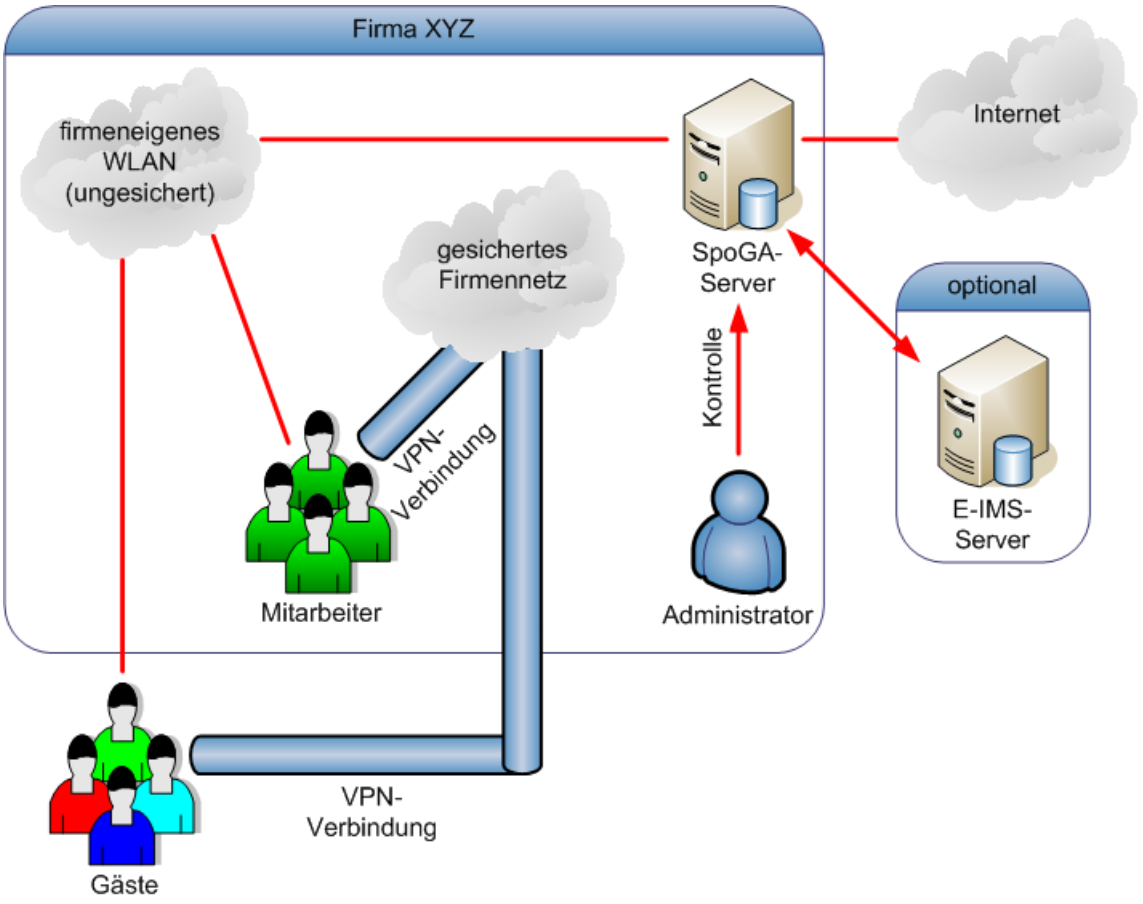


Abbildung 3.4: Architektur des zu entwickelnden Systems

4 Groupware

Die Basis des E-IMS-Systems bildet eine Groupware-Software. Diese erleichtert die Programmierarbeit, da einige hilfreiche Module (z.B. ein Registrierungs-System, ein Terminkalender usw.) bereits integriert sind und somit nicht neu zu entwickeln, sondern lediglich anzupassen und zu ergänzen waren. Als Team-Arbeit des Gesamt-Projektes wurden dabei drei häufig eingesetzte Groupware-Systeme ausgewählt und getestet:

Tester	**Name der Groupware**	**Referenz**
Christoph Speich	phpGroupware	[Spe07]
Markus Müller	OpenXChange	[Mül07]
Matthias Ehrenstein	OpenGroupware	Kapitel 4.1 dieser Arbeit

Tabelle 4.1: Übersicht der getesteten Groupware-Systeme

Diese Arbeit beschäftigt sich dabei mit dem Testen der OpenGroupware. Als mögliche Alternative zu OpenGroupware hat sich Markus Müller mit der Groupware OpenXChange und Christoph Speich mit phpGroupware beschäftigt. Die jeweiligen Resumees sind in deren Studienarbeiten (siehe Tabelle 4.1) nachzulesen.

Des Weiteren wurde ein Interview mit Herrn Uwe Arndt, wissenschaftlicher Leiter des Rechenzentrums der Universität Koblenz geführt, der sich aktuell ebenfalls mit dem Testen von Groupware-Systemen befasst. (Siehe Anhang B)

SpoGA, in Verbindung mit der E-IMS-Erweiterung und somit Nutzung der Groupware, stellt einen Mehrnutzen für die Veranstalter dar. Es ist eine größere Teilnehmer- bzw. Nutzerzahl verwaltbar und weitere Funktionen wie z.B. eine Kostenabrechnung der Nutzung sowie eine Ressourcenplanung sind integriert.

4.1 Auswahl der Software

Bei der Auswahl wurde in dieser Ausarbeitung das von der Skyrix Software AG entwickelte und seit 2003 als Open Source verfügbare Produkt „OpenGroupware.org“ betrachtet. Dies ist eine Server-Applikation, die von Anwendern im Web-Browser bedient wird. Neben der Kernanwendung, die in Objective C geschrieben wurde, setzt OpenGroupware auf bewährte Standardkomponenten wie PostgreSQL (Datenbankmanagementsystem) und Apache (Webserver). OpenGroupware integriert Funktionen wie Terminplaner, Kontakt-Datenbank für Personen und Unternehmen, Ressourcen- sowie Projekt- und Aufgabenverwaltung, Webmail-Client und ein einfaches News-System für Intranets. Zudem ist ein Abgleich mit Palm-PDAs möglich.

Zitat von OpenGroupware.org [Ope]:

> *„Mission: To create, as a community, the leading open source groupware server to integrate with the leading open source office suite products and all the leading groupware clients running across all major platforms, and to provide access to all functionality and data through open XML-based interfaces and APIs.“*

Die Mission von OpenGroupware ist eindeutig: Es stellt ein führendes Groupware-System mit integrierten Open Source Office-Produken dar, das Plattform-unabhängig und durch Schnittstellen und APIs leicht zugänglich ist.

Das OpenGroupware-Team sieht die eigene Software als Ersatz für Exchange von Microsoft. Die Software läuft unter Linux, und übliche Clients wie Microsoft Outlook, Ximian Evolution, der Mozilla-Kalender, der Kalender iCal von Apple und der OpenOffice-Client Glow werden unterstützt. Der zukünftige Support soll durch *kostenpflichtige Zusatzmodule* gesichert sein. Dazu zählt z.B. Zide Store, ein Plug-In, das Mail-Clients wie Outlook oder Evolution einbindet, und DocShare, eine Software, die OpenGroupware um Funktionen zum Dokumentenaustausch erweitert.

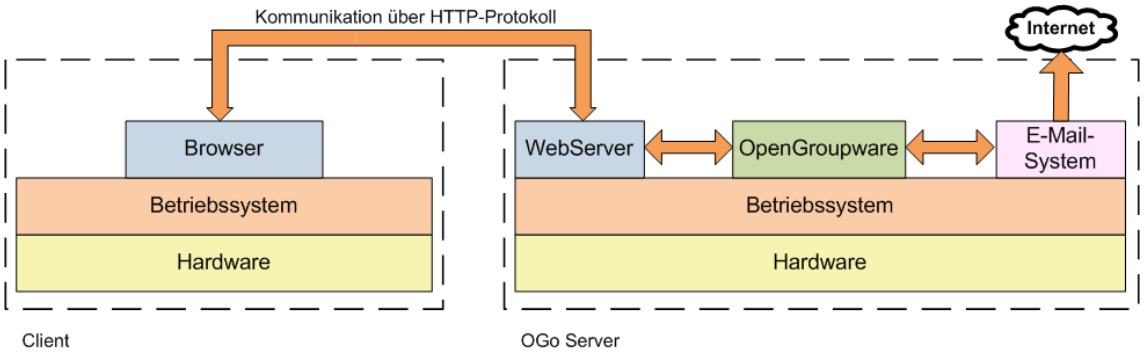


Abbildung 4.1: Kommunikation zwischen OpenGroupware Server und Client (Quelle: in Anlehnung an [Ope])

In Abbildung 4.1 ist die Struktur der Kommunikation zwischen Server und Client dargestellt. Ein wesentliches Merkmal von OpenGroupware ist hierbei, dass die Clients keinerlei eigene Software benötigen. Ein herkömmlicher Web-Browser reicht aus, um die Funktionen zu nutzen. Dieser greift mittels HTTP-Protokoll auf den Apache-Webserver zu, welcher die XML Seiten zur Verfügung stellt. Ein E-Mail System ermöglicht den Versand bzw. Empfang von E-Mails. Für das E-IMS-System ist lediglich der Versand nötig. Daher sind keine kostenpflichtigen E-Mail Plugins für den Empfang von Kunden-Nachrichten erforderlich.

4.2 Testen der Software

Es gibt verschiedene Arten, OpenGroupware zu testen

- **Eine Knoppix Boot-CD** [1]

  Wahlweise kann man hier entweder eine CD für 5 Euro + Versand bestellen, oder sich ein ca. 560 MB großes ISO-Image herunterladen. Nach Brennen des Images kann man dann von CD booten und es wird ein Knoppix-System gestartet, unter dem bereits OpenGroupware installiert ist. Nach Starten eines Web-Browsers gelangt man dann auf die Startseite des OpenGroupware-Systems.

- **Eine Online-Demo von Skyrix** [2]

  Nach der Registrierung bekommt man einen Link zur Online-Demo und zwei Logins (Benutzername + Passwort) für die Nutzung. Als Test-Datenbestand sind schon einige Projekte und Termine eingetragen, mit denen man arbeiten kann. Dies ermöglicht das sofortige und unkomplizierte Testen der Software, ohne zeitaufwendige und (nach eigenen Angaben von opengroupware.org) nicht ganz einfache Installation unter Linux. Der Test-Zugang ist voll funktionsfähig und reicht aus, um sich ein Bild von OpenGroupware zu machen.

- **In ein vorhandenes Linux-System einbinden**

  Alternativ kann man die Software auch per Paketmanager (APT[3]) in eine bestehende Linux-Distribution wie z.B. Debian oder SuSE integrieren.

4.3 Bewertung

Um OpenGroupware zu bewerten, orientierten sich die Tester nach der *ISO 9126 (DIN 66272)-Norm* [Wik]. Hierbei wird anhand von 6 Haupt-Bewertungspunkten eine Beurteilung von Software vorgenommen. Bezogen auf OpenGroupware ergab sich dabei folgendes Resultat:

- **Funktionalität**

  Kernfrage: „Inwieweit besitzt die Software die geforderten Funktionen?“

  Als Anforderungen an das Basissystem des E-IMS-Servers haben sich die Tester auf folgende Punkte geeinigt:

  - Verwaltung von Personen und Unternehmen
  - E-Mail System (nur Versand nötig)
  - Einladungs-System (zum Einladen von Usern zu Meetings oder ähnlichem)
  - Ressourcenverwaltung (z.B. Beamer, Räume, PCs)
  - Dokumentmanager (zur gemeinsamen Nutzung von Dokumenten)
  - Kalenderfunktion

[1] Knoppix: http://www.opengroupware.org/de/knoppix/index.html
[2] Skyrix Demo: http://www.skyrix.de/de/forms/onlinedemo.php
[3] Advanced Package Tool, Paketmanager von Linux

- Terminplanung/verwaltung
- Newsportal (Anzeige von wichtigen Informationen)
- Synchronisation (z.B. von mobilen Endgeräten)

Fast alle Funktionen werden von OpenGroupware bereitgestellt. Ein für dieses Projekt zu vernachlässigender Nachteil ist, dass das MS Outlook Plug-In kostenpflichtig ist, wie oben bereits erwähnt wurde, dieses aber nicht benötigt wird. Ein Einladungssystem, um Gäste zu einer Veranstaltung einzuladen, ist ebenfalls nicht vorhanden und müsste implementiert werden.

- **Zuverlässigkeit**

  Kernfrage: „Kann die Software ein bestimmtes Leistungsniveau unter bestimmten Bedingungen über einen bestimmten Zeitraum aufrechterhalten?“

  Die Software läuft komplett auf einem Server. Dieser sollte möglichst leistungsfähig und sicher gegen externe Angriffe sein, damit die Sicherheit und Zuverlässigkeit des Systems gewährleistet ist. Es ist sinnvoll, eine aktuelle Linux-Distribution mit Minimalinstallation, und einem RAID-Verbund zu wählen. Desweiteren sollte ein Backup-System (DVD oder Tape) benutzt werden. Um möglichst viele Nutzeranfragen zur gleichen Zeit verarbeiten zu können, sollte die Internetanbindung auch möglichst breitbandig sein. Werden diese Punkte berücksichtigt, so bietet OpenGroupware eine zuverlässige Groupware-Lösung.

- **Benutzbarkeit**

  Kernfrage: „Welchen Aufwand fordert der Einsatz der Software von den Benutzern und wie wird er von diesen beurteilt?“

  Die Bedienung erfolgt plattformunabhängig über einen Web-Browser. Um einzelne Funktionen wie z.B. die Terminplanung oder den Kalender zu nutzen, kann man alternativ über verschiedene Client-Software (Microsoft Outlook, Ximian Evolution, den Mozilla-Kalender, den Kalender iCal von Apple und den OpenOffice-Client Glow) auf den Server zugreifen. Im Gegensatz zu vergleichbaren Lösungen sind die technischen Anforderungen der OpenGroupware Software sehr gering. Sie verzichtet zum Beispiel auf Java-Applets und lässt sich ohne JavaScript betreiben. Jegliche Installation von Zusatzsoftware auf dem Clientsystem entfällt. Das Menu ist klar strukturiert, und auch ein unversierter Nutzer hat sich schnell in die verschiedenen Funktionen eingearbeitet. Die Benutzeroberfläche ist weitestgehend selbsterklärend.

- **Effizienz**

  Kernfrage: „Wie liegt das Verhältnis zwischen Leistungsniveau der Software und eingesetzten Betriebsmitteln?“

  Es wird keine spezielle Hard- oder Software für OpenGroupware benötigt. Ein üblicher Web- bzw. Datenbankserver, der in jedem Unternehmen steht, reicht aus. Ein weiterer, entscheidender Punkt ist, dass OpenGroupware (bis auf das Outlook Plug-In) völlig kostenlos ist. Somit kann man problemlos ein ganzes Unternehmen mit dieser Lösung ausstatten, ohne hohe Kosten für Lizenzen zu verursachen.

- **Änderbarkeit**

Kernfrage: „Welchen Aufwand erfordert die Durchführung vorgegebener Änderungen an der Software?“

OpenGroupware ist seit 2003 Open Source, und wird seitdem permanent weiterentwickelt. Es existieren zahlreiche Foren, Mailinglists und sogar ein IRC-Channel, wo sich Entwickler über mögliche Probleme beraten und austauschen können. Als Beispiel für ein gutes deutschsprachiges Forum ist de.opengroupware.info zu nennen. Weitere Informationen dazu findet man auf der offiziellen Webseite [Ope] unter der Rubrik „Support“. Für geübte Programmierer ist es somit sehr einfach das System zu erweitern bzw. Änderungen einzuarbeiten.

- **Übertragbarkeit**

  Kernfrage: „Wie leicht lässt sich die Software in eine andere Umgebung übertragen?“

  Das System ist an eine Linux-Distribution gebunden. Um den Server in einer Windows-Umgebung zu betreiben, muss man auf Emulatoren wie CYGWIN zurückgreifen. Aufgrund der XML API ist OpenGroupware leicht in ein bestehendes System zu integrieren.

Nach abschließender Diskussion und Auswertung der Tests wurde sich für eGroupware als Basis des E-IMS-Systems entschieden. Dies ist eine Variante von phpGroupware und scheint den Testern am besten geeignet für dieses Projekt, da es auf PHP basiert und durch seinen modularen Aufbau leicht zu erweitern bzw. anzupassen ist. Weitere Argumente, die zur Nutzung von eGroupware geführt haben, findet man in den Arbeiten von Christoph Speich ([Spe07]) und Markus Müller ([Mül07]).

5 Installation des Servers

5.1 Installation des Betriebssystems

Als Basis für den SpoGA-Server dient eine Minimal-Installation von SuSE Linux 10.0[1]. Diese Distribution von Linux hat den Vorteil, dass sie in der Minimal-Installation bereits viele für den SpoGA-Server benötigten Komponenten mitbringt. Ein weiterer, entscheidender Vorteil, z.B. gegenüber Microsoft Windows XP, sind die vergleichsweise geringen Hardware-Anforderungen und teure Lizenskosten entfallen ebenfalls. Eine grafische Oberfläche wird nicht benötigt, wodurch sich nochmals einige System-Ressourcen einsparen lassen. Als Testsystem dient ein PC mit einem Intel Celeron Prozessor (500 Mhz) und 384 MB RAM. Dies erwies sich für Testzwecke in Laborumgebung als völlig ausreichend.

Bei den Paketen, die zusätzlich zu installieren sind, handelt es sich im Einzelnen um:

- MySQL (Datenbankserver)
- Apache mit php-mod (Webserver)
- GCC (C/C++-Compiler)

Weitere Pakete die den Umgang mit dem System erleichtern:

- OpenSSH (SSH-Verbindungssuite)
- MC (Dateimanager)
- JOE (Texteditor)

Ich verzichte an dieser Stelle bewusst darauf, die einzelnen Schritte einer Linux Installation detailliert zu beschreiben, da es zu diesem Thema bereits ausreichende Dokumentationen und Handbücher gibt ([Nov]).

Um den Server gegen externe Angriffe zu sichern, wird die Linux-interne Firewall *iptables*[2] benutzt. Alle Ports bis auf SSH, HTTP und HTTPS werden gesperrt. (Detailliertere Informationen erfolgen in Kapitel 7.5)

5.2 Installation der benötigten Software

Zunächst wird eine Minimal-Installation von SuSE Linux 10.0 installiert. Diese hat folgende Vorteile gegenüber einer normalen Installation:

- **Leistungsfaktor:**
  Die Minimal-Installation hat nur geringe Hardware-Anforderungen, da z.B. keine grafische Oberfläche installiert wird.

[1] SUSE Linux 10.0: http://www.novell.com/products/suselinux/
[2] iptables: http://www.netfilter.org/projects/iptables/index.html

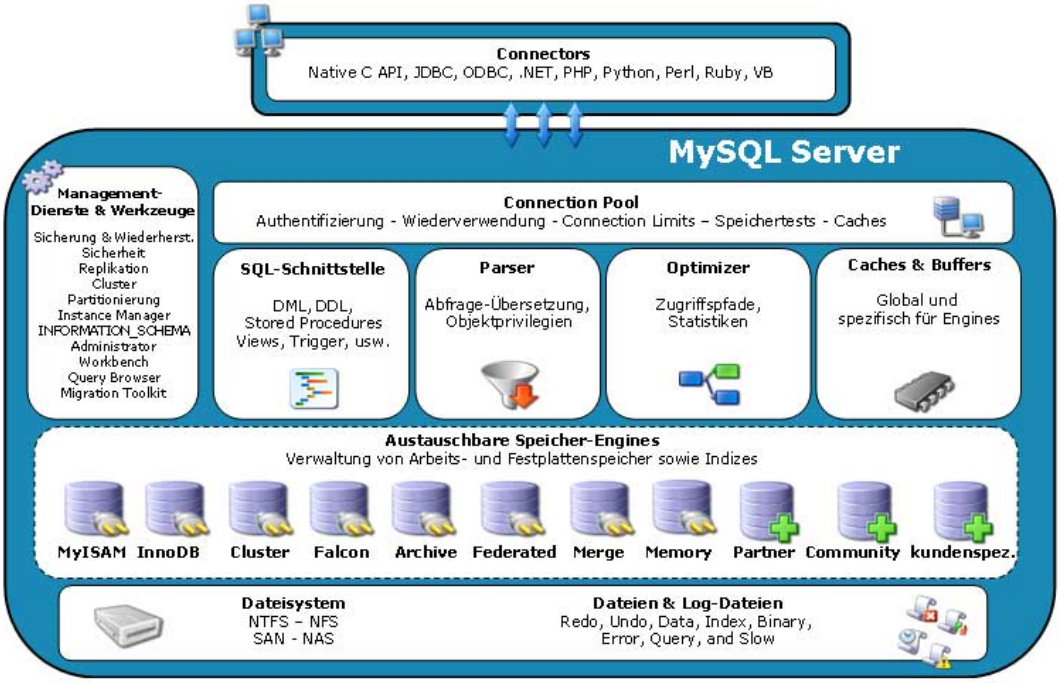


Abbildung 5.1: Übersicht MySQL-Server 5.0 (Quelle: [MyS])

- **Sicherheitsfaktor:**
  nur die nötigsten Programme werden installiert und somit sind mögliche Angriffspunkte minimal.

- **Zeitfaktor:**
  Die Minimal-Installation ist in weniger als einer Stunde (abhängig von der Systemleistung) einsatzbereit.

Nun werden noch die zusätzlich benötigten Pakete (MySQL-Server, Apache-Server, GCC, OpenSSH, MC, JOE) installiert. Diese erfüllen, kurz zusammengefasst, folgende Aufgaben:

- **MySQL-Server**
  Datenbank-Server haben die Aufgabe, beim Betrieb eines Datenbanksystems die Daten so zu organisieren, dass eine effektive Datenspeicherung möglich ist. Dabei sollte eine möglichst hohe Stabilität und Performance gewährleistet sein.

  Als Datenbank-Server-Software für dieses Projekt dient die weit verbreitete und bekannte Open-Source-Lösung MySQL des schwedischen Herstellers MySQL AB ([MyS]). Es gibt weit umfangreichere und komplexere Produkte, aber für den Einsatz mit SpoGA ist MySQL völlig ausreichend. Die bekanntesten Open-Source Alternativen wären zum Beispiel postgresSQL, MaxDB, Firebird oder Ingres. An dieser Stelle soll nicht näher auf den Vergleich dieser Datenbank-Management-Systeme (kurz DBMS) eingegangen werden. Weiter Informationen sind einem Artikel von Heise ([Hor06]) zu entnehmen.

Die zur Zeit aktuelle Version ist MySQL 5.0, welche auch für SpoGA zum Einsatz kommt. In Abbildung 5.1 sind die wichtigsten Merkmale grafisch dargestellt. Als „connector“ (also die Programmierschnittstelle zu anderen Anwendungen (API)) dient für dieses Projekt PHP. Zur Administration und Konfiguration des MySQL Servers wird das ebenfalls weit verbreitete Open-Source Produkt phpMyAdmin ([phpa]) auf Basis von PHP verwendet.

- **Apache-Webserver**
  Der Apache Webserver ist laut der Statistik von Netcraft[3] der zur Zeit am häufigsten genutze Webserver der Welt. Das Apache Projekt ist heute, neben Linux selbst, das erfolgreichste und bekannteste Open-Source-Projekt. Es entstand Mitte der 90er Jahre aus dem NCSA[4], dem damals beliebtesten Webserver. Da dessen Entwicklung jedoch eingestellt wurde, gründete man daraufhin die Apache Group.

  Die wesentlichen Merkmale von Apache sind:

  – Open-Source
  – Unterstützung von verschiedenen Betriebssystemen (Linux, Windows, MacOS,...)
  – modularer Aufbau, dadurch individuell einsetzbar und leicht zu erweitern
  – kostenlos

- **GCC**
  GCC, die GNU Compiler Collection, enthält Compiler und Laufzeit-Bibliotheken für C++, C, Objective C, Fortran, Java und Ada. Er unterstützt hunderte von Kombinationen von Prozessor, Rechnerarchitektur und Betriebssystem, darunter alle von Linux und den BSD-Systemen unterstützte Architekturen. Auf dem Testsystem kommt GCC in der Version 4.0.2 zum Einsatz.

- **OpenSSH**
  OpenSSH ist eine freie Version der SSH-Verbindungssuite. OpenSSH verschlüsselt die Passwörter und die gesamte Verbindung, um Mithören (eavesdropping), das Entführen von Verbindungen (connection hijacking) und ähnliche Angriffe zu unterbinden. Zusätzlich bietet OpenSSH neben einer sicheren Authentifikationsmöglichkeit die Unterstützung für alle SSH-Protokollversionen an. Für das Testsystem wird die Version 4.1p1 verwendet, um Dateien sicher auf den Server zu übertragen.

- **MC**
  Der Midnight Commander (MC) ist ein Dateimanager für die Linux-Konsole. Er ist vergleichbar mit dem sehr populären Norton-Commander für DOS und stellt eine sehr angenehme Arbeitserleichterung dar. Einer seiner größten Vorteile ist seine Vielfalt: Maus-Unterstützung (GPM), eingebauter FTP-Client, Entpacken von sämtlichen Archiven und Paketen, normale Operation wie Kopieren und Verschieben von Dateien und vieles mehr.

- **JOE**
  JOE ist ein Akronym und steht für „Joe's Own Editor“. Dies ist ein weit verbreiteter

[3] Netcraft Statistiken: http://news.netcraft.com
[4] NCSA Webserver: Ein vom National Center for Supercomputing Applications entwickelter Webserver, vgl. http://hoohoo.ncsa.uiuc.edu/index.html

und einfach zu bedienender Texteditor für Linux. Er unterstützt unter anderem auch Syntax-Highlighting, also die farbliche Hervorhebung von Wörtern oder Text-Passagen. Dies ist besonders bei der Programmierung von PHP oder C++ sehr hilfreich. Verwendet wird JOE in der Version 3.3.

5.3 Anpassung des Systems

Nachdem die nötige Software für den SpoGA-Server installiert ist, sind nun einige Anpassungen des Systems nötig. Da der SpoGA-Server ja als Router zwischen Firmennetz/Internet und WLAN-Netz dienen soll, ist die Installation einer 2. Netzwerkkarte erforderlich. Nach dessen Einbau wird die Karte über YAST2 konfiguriert. Dabei wird eine statische IP-Adresse (hier 192.168.10.1) mit Subnetzmaske 255.255.255.0 vergeben. Es handelt sich dabei um ein privates Class-C Netzwerk, was mit 253 nutzbaren IP-Adressen (Adresse 0 und 255 sind für Netzmaske sowie Broadcast-Adresse belegt) völlig ausreichend ist. In Abbildung 5.2 ist die Netzwerk-Konfiguration des SpoGA-Servers schematisch dargestellt.

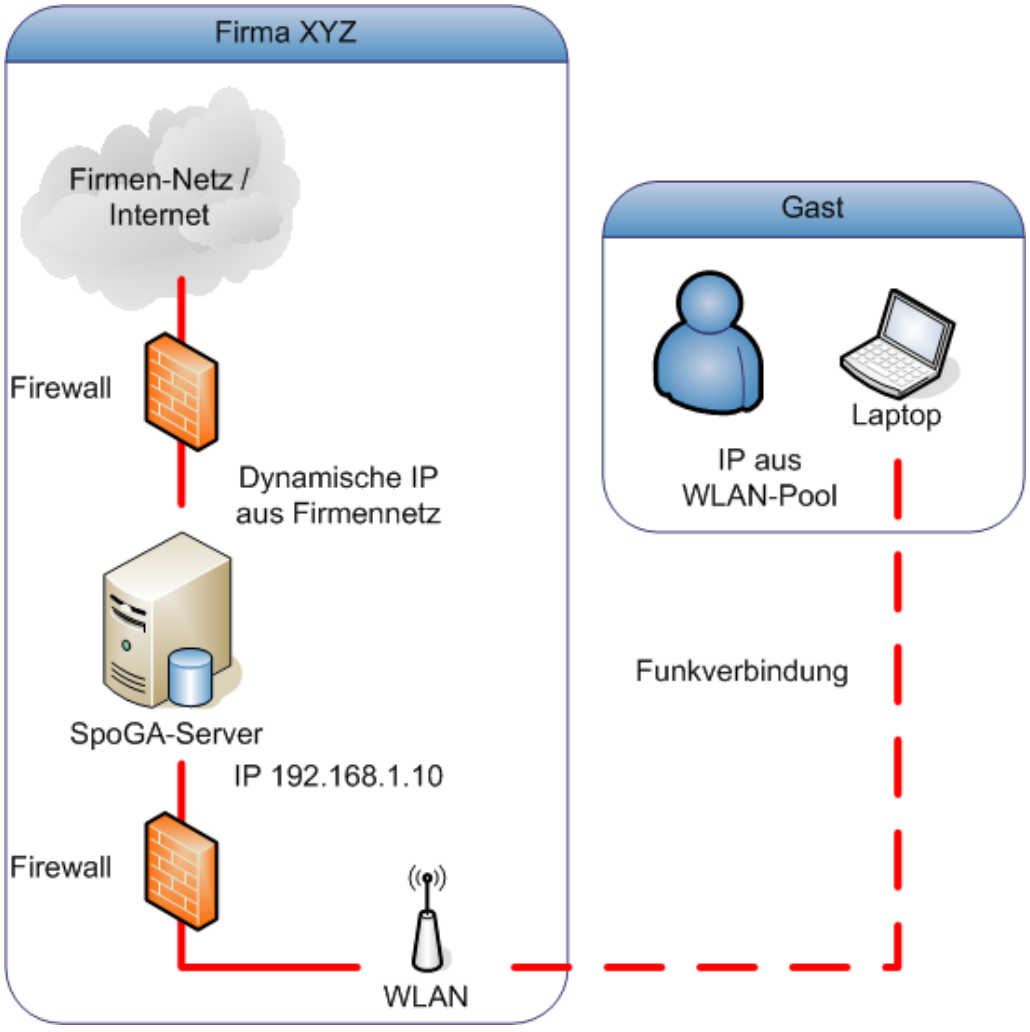


Abbildung 5.2: Übersicht der Netzwerk-Konfiguration des SpoGA-Servers

6 Bedienung SpoGA

6.1 Einleitung

Dieses Kapitel stellt zunächst die für die Benutzeroberfläche verwendete Skript-Sprache PHP vor. Anschließend folgt eine Zusammenfassung der Möglichkeiten, die ein Benutzer hat, um das SpoGA-System zu bedienen. Es wird unter anderem vorgestellt, wie man

- als Administrator Gastgeber freischaltet,
- als Gastgeber Gäste freischaltet,
- sich als Gast nach Erhalt des Freischalt-Codes am System an- und auch wieder abmeldet,
- als Gast seinen aktuellen Status und die Restnutzungsdauer abruft,
- als Gast seinen Code erneut anfordert, falls man ihn vergessen oder verlegt hat

Es soll nicht jeder Punkt detailliert geschildert, sondern lediglich eine grobe, funktionale Übersicht vermittelt werden.

6.2 Die Benutzeroberfläche

Die Benutzeroberfläche stellt die Verbindung zwischen Server und Anwender her. Es handelt sich dabei um ein Web-Frontend, das es dem Nutzer ermöglicht, sich am SpoGA-System anzumelden. Für administrative Zwecke stehen noch weitere Funktionen zur Verfügung.

6.2.1 Die Sprache

Als Grundgerüst für die Benutzeroberfläche dient eine dynamische Webseite, die mit der Skript-Sprache PHP[1] umgesetzt ist. Die Abkürzung steht für *PHP: Hypertext Preprocessor*, eine weitverbreitete Open-Source-Skriptsprache speziell für Webentwicklungen. Es läßt sich leicht in HTML einbinden und die Syntax erinnert ein wenig an C, Java und Perl. Das Hauptziel dieser Sprache ist es, Webentwicklern die Möglichkeit zu geben, schnell dynamisch generierte Webseiten zu erzeugen. Im Gegensatz zu einer Client-seitigen Sprache wie Javaskript, wird PHP Server-seitig ausgeführt. Der ursprüngliche Source-Code bleibt für den Webseitenbesucher nicht ersichtlich.

Die wesentlichen Merkmale von PHP sind (Quelle: [Ber05]):

- Serverseitige Skript-Sprache

[1] PHP: http://www.php.net/

- Erzeugung von dynamischen Webseiten
- leicht zu erlernen
- riesiger Funktionsumfang
- sowohl für Einsteiger als auch für Webentwickler geeignet
- Plattformunabhängig (Unix, Windows, Mac OS, ...)
- Unterstützung für eine breite Masse an Datenbanksystemen

Für weiterführende Informationen zu PHP stehen dem interessierten Leser zahlreiche Dokumentationen, Handbücher und Tutorials unter anderem in dieser Linksammlung[2] zur Verfügung.

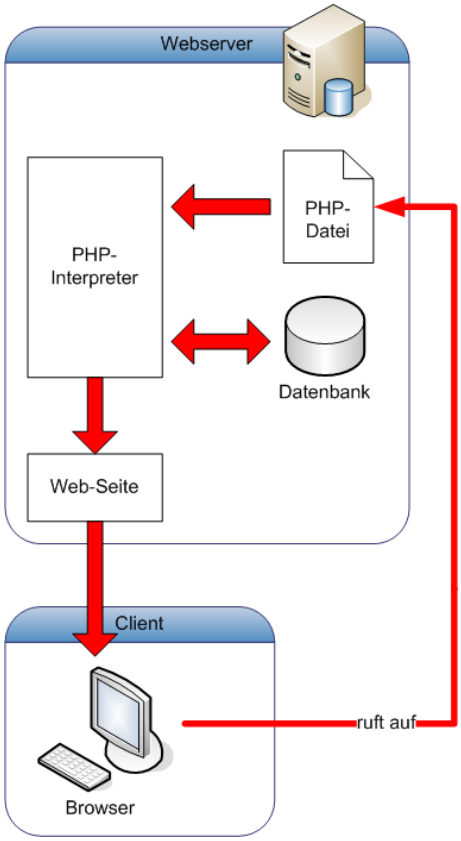


Abbildung 6.1: Zusammenspiel von PHP, Webserver und Datenbanksystem

Als Datenbanksystem für das SpoGA-System dient, wie bereits zuvor beschrieben, MySQL[3]. Der Apache-Webserver beinhaltet den PHP-Interpreter und es wird keine weitere Software für die Erstellung der Webseiten benötigt. In Abbildung 6.1 ist das grundlegende Zusammenspiel von Webserver, PHP-Interpreter und Datenbanksystem gut dargestellt.

[2]PHP-Linksammlung: `http://www.gdu.de/links`
[3]MySQL: `http://www.mysql.com/`

Der Benutzer greift mit seinem Browser auf die PHP-Datei zu, die auf dem Webserver liegt. Der Interpreter, in diesem Fall die Webserver-Software Apache2, generiert aus den Informationen der PHP-Datei eine Webseite. Diese Webseite wird dann an den Browser des Benutzer-PCs (Client) geschickt und kann von dort betrachtet werden. Der Interpreter greift dabei, abhängig vom Inhalt der PHP-Datei, auf eine Datenbank zu und holt sich die vom Benutzer angeforderten Daten. Diese Datenbank liegt ebenfalls auf dem Webserver und beinhaltet alle notwendigen SpoGA-Daten, wie z.B. Handynummer des Benutzers, Nutzungszeit, Informationen über Zeit und Dauer seiner Aktivitäten usw. Mehr dazu in den folgenden Kapiteln.

6.2.2 Die SpoGA-Startseite

Abbildung 6.2: SpoGA Startseite

Standardmäßig gelangt man nach einem erfolgreichen Verbindungsaufbau zum SpoGA-Server über einen Internet-Browser auf die Startseite (oder auch Index-Seite genannt; hier: index.php). Ist ein Benutzer angemeldet, so befindet sich auf dieser Startseite ein individueller Begrüßungstext, sowie der derzeitige Status des Benutzers (aktuelle IP-Adresse, Handynummer, Kostenträger und verbleibende sowie bereits „versurfte“ Nutzungszeit).

Die Startseite bildet das Zentrum der Benutzeroberfläche. Die Navigation zu den Funktionen des SpoGA-Systems erfolgt über das Top-Menu im oberen Bereich der Seite. Von hier aus kann man sich am System an- bzw. abmelden, einen gültigen Code erneut anfordern oder mit entsprechender Autorisierung Gastgeber bzw. Gäste hinzufügen/löschen.

6.2.3 Der Administrator-Bereich

Abbildung 6.3: Administrator-Bereich

Der Administrator-Bereich (index-admin.php) ist vom Administrator selbst oder von Gastgebern nutzbar. Das SpoGA-System entscheidet bei Eingabe von Benutzername und Kennwort in diesem Bereich, ob es sich um einen Administrator oder einen Gastgeber handelt.

Gäste/Gastgeber verwalten

Ein Administrator hat die Möglichkeit Gastgeber zu verwalten. Er kann diese hinzufügen und auch wieder löschen, sowie sich eine Liste der bereits im System vorhandenen Gastgeber-Zugänge ansehen. Vorraussetzung dazu ist ein gültiger Administrator-Zugang, welcher in der Konfigurationsdatei jederzeit verändert werden kann. Zudem verfügt er über die selben Rechte wie ein Gastgeber.

Wurde von einem Administrator erfolgreich ein Gastzugang angelegt, so besteht für den Gastgeber die Möglichkeit einzelne Gastzugänge einzurichten und zu verwalten. Um Gäste hinzuzufügen sind folgende Angaben nötig (siehe dazu Abbildung 6.4):

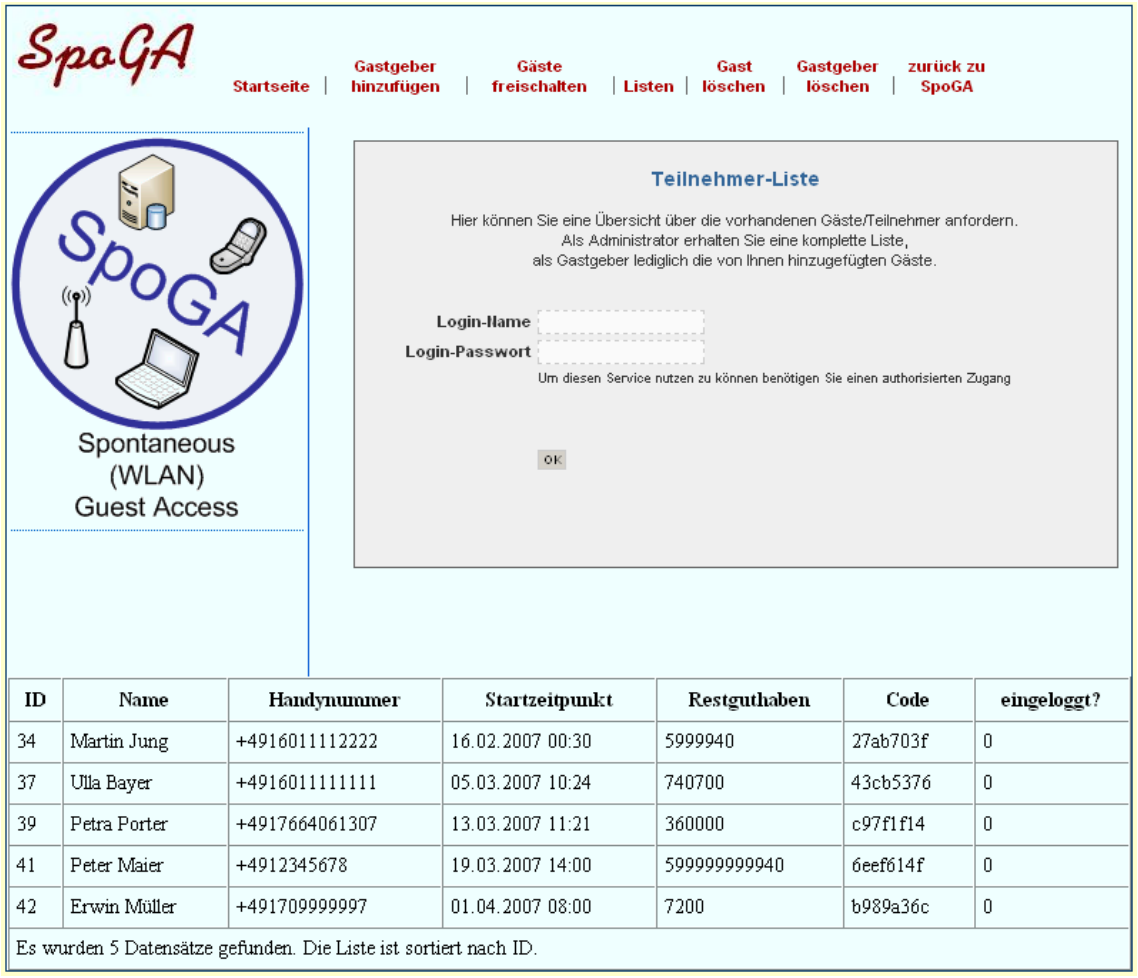
- Name des Gastes
- Handynummer des Gastes
- Startzeitpunkt, ab dem der Gastzugang gültig werden soll (Angabe von Datum und Uhrzeit)
- Zeitgutschrift in Minuten, wie lange der Gastzugang ab Startzeitpunkt gültig ist

Nach Anlegen des Gast-Zugangs, wird eine Stunde vor Startzeitpunkt ein Code generiert, welcher für den Gast zum Anmeldem am System benötigt wird, und per SMS an die Handynummes des Gastes gesendet. Ist die Zeitgutschrift abgelaufen, so verliert der Code seine Gültigkeit und der Gastzugang wird inaktiv.

Abbildung 6.4: Gäste hinzufügen

Listen

Um eine Übersicht der Teilnehmer zu erhalten und zu Kontrollzwecken haben die Gastgeber und der Administrator eine *Listen-Funktion*. Hier wird nach Eingabe des Gastgeber- bwz. Administratorkennung eine Liste aller Teilnehmer (nur für Administrator) oder der Teilnehmer, die diesem Gastgeber zugeordnet sind, angezeigt. Zusätzlich kann sich ein Administrator eine Liste aller von ihm angelegten Gastgeber anzeigen lassen.



ID	Name	Handynummer	Startzeitpunkt	Restguthaben	Code	eingeloggt?
34	Martin Jung	+4916011112222	16.02.2007 00:30	5999940	27ab703f	0
37	Ulla Bayer	+4916011111111	05.03.2007 10:24	740700	43cb5376	0
39	Petra Porter	+4917664061307	13.03.2007 11:21	360000	c97f1f14	0
41	Peter Maier	+4912345678	19.03.2007 14:00	599999999940	6eef614f	0
42	Erwin Müller	+491709999997	01.04.2007 08:00	7200	b989a36c	0

Es wurden 5 Datensätze gefunden. Die Liste ist sortiert nach ID.

Abbildung 6.5: Teilnehmer-Listen

6.2.4 Der Gast-Bereich

Anmelden

Um sich am System anmelden zu können, benötigt ein Gast einen gültigen 8-stelligen Code. Diesen hat er per SMS kurz vor Veranstaltungsbeginn (standardmäßig eine Stunde vorher) erhalten. Genauere Informationen zu Generierung und Versand des Codes sind im Kapitel 7.2 nachzulesen.

Diesen trägt der Gast nun in das Textfeld der Anmelde-Seite ein und bestätigt mit OK. Daraufhin erfolgt eine Überprüfung des Codes in der SQL-Datenbank und es erfolgt entweder eine Erfolgsmeldung „Sie sind nun angmeldet und können lossurfen“ oder eine von fünf verschiedenen Fehlermeldungen:

- Fehler beim Datenbankzugriff (MySQL)
  Hier ist etwas beim Zugriff auf die Datenbank nicht in Ordnung. Entweder eine defekte Datenbank oder ein falsches Passwort.

- Code falsch oder Account existiert nicht! Überprüfen Sie Ihre Eingabe.
  Sofern der eingegebene Code nicht in der Datenbank vorhanden.
- Ihnen steht keine Nutzungszeit mehr zur Verfügung!
  Die begrenzte Nutzungszeit des Codes ist abgelaufen und nicht mehr gültig.
- Es existiert kein aktuell gültiger Zugang für Sie. Sie können sich nur innerhalb Ihrer Veranstaltung anmelden.
  Die Veranstaltung ist vorrüber und der Code abgelaufen.

Abbildung 6.6: Anmeldeseite

Nach erfolgreicher Anmeldung wird der Benutzer in der Firewall freigeschaltet und ist somit berechtigt, das Firmen-Wlan zu nutzen. Nun beginnt die Nutzungszeit zu laufen bzw. weiterzulaufen, falls der Gast zu einem früheren Zeitpunkt bereits angemeldet war. Ebenso wird die noch verbleibende Restnutzungdauer (Credits) verringert. Zu Kontrollzwecken der Benutzer-Aktivität, sowie zu einer genaueren Abrechnungsmöglichkeit durch das optionale E-IMS-Billing, wird jeder An- sowie Abmeldevorgang mit den Daten des Benutzers (IP, Zeit, Dauer, Handynr., etc) in einer Log-Tabelle der Datenbank protokolliert. Die Kommunikation im WLAN erfolgt dabei unverschlüsselt. Wenn ein Gast eine sichere Verbindung zum Firmen-Netz herstellen will, so sollte er dazu eine VPN-Verbindung nutzen.

Abmelden

Nach Ablauf der Nutzungszeit erfolgt die automatische Abmeldung vom System. Dabei wird der Benutzer von der Firewall wieder geblockt (vgl. 7.5), und erreicht somit nur noch die Startseite. Momentan ist für solch eine Situation keine Warnmeldung eingebaut, da es in dem zeitlich begrenzten Rahmen dieser Arbeit technich nicht umzusetzen war.

Es besteht zudem die Möglichkeit der manuellen Abmeldung durch die Abmelde-Seite. Dies kann sinnvoll sein, wenn ein Gast einen kostenpflichtigen Zugang nutz und sich während einer Pause abmelden will. Die Nutzungszeit wird dabei angehalten und die Restnutzungsdauer aktualisiert.

Freischalt-Code anfordern

Hat ein Benutzer einmal aus Versehen die SMS mit den SpoGA-Anmeldedaten gelöscht, oder will er ihn aus sonstigen Gründen erneut anfordern, so hat er mit dieser Seite die Möglichkeit dazu. Nach Eingabe seiner Handynummer im internationalen Format (+49175..) wird der Code, der zu dieser Nummer passt und wozu eine gültiger Gastzugang existiert, erneut per SMS an den Benutzer gesendet. Fehler werden mit folgenden Meldungen abgefangen:

- Bitte gültiges internat. Format (zB +49171...) benutzen!
  Falsches Format der Eingabe. Es wird ein internationales Format gefordert.
- Ihre Nutzungsdauer ist abgelaufen! Wenden Sie sich bitte an Ihren Gastgeber
- Es existiert momentan kein Gastzugang für Sie. Falls doch, wenden Sie sich bitte an den Gastgeber.
- Handynummer falsch oder Account existiert nicht! Wenden Sie sich bitte an Ihren Gastgeber
  Wenn die Handynummer trotz gültigem Format nicht zum Code passt. Dies verhindert den Missbrauch durch Weitergabe des Codes.

Sind die Angaben korrekt und existiert ein gültiger Gast-Zungang zu der angegeben Nummer, so wird eine Nachricht mit dem entsprechenden 8-stelligen Code und zusätzlichen Informationen zum Kostenträger der Nutzung an den Benutzer gesendet. Mögliche Kostenträger sind der Gastgeber (im Falle des manuellen Gastzugangs), der Benutzer selbst, oder die kostenfreie Nutzung (beispielsweise bei geladenen VIP-Gästen).

6.3 Die Logik

Zur Überwachung des Systems läuft auf dem SpoGA-Server permanent ein C++ Programm. Dieses kontrolliert die Aktivitäten der Benutzer und ist für das rechtzeitige Senden der SMS sowie das Füllen der log-Tabelle zu Kontrollzwecken zuständig. Ebenso wird, wenn ein Nutzer eingeloggt ist, im 10 Sekunden Takt seine Rest-Nutzungszeit überprüft und verringert. Ist diese abgelaufen oder meldet er sich manuell vom System ab, so wird die Verbindung des Benutzers zum Internet per Firewall beblockt. Gleichzeitig wird die Gesamt-Nutzungszeit erhöht und in der log-Tabelle für spätere Auswertungen (z.B. Billing) festgehalten . Eine Stunde vor Veranstaltungs-Beginn wird über den SMS-Gateway automatisch eine SMS mit dem Freischalt-Code an den Benutzer gesendet. Genauere Informationen zum C++-Programm sowie den SMS-Versand sind im Kapitel 7 nachzulesen.

6.4 Die Kontrolle

Für spätere Auswertungen und zur besseren Nutzerkontrolle wird durch das C++ Programm eine log-Tabelle der SQL-Datenbank gefüllt. Sobald sich ein Benutzer am System anmeldet, erfolgt ein neuer Log-Eintrag. Festgehalten werden dabei: Handynummer, Beginn- und Endzeit seiner Sitzung als Unix-Timestamp, die Dauer der Sitzung in Sekunden sowie eine ID zur Bestimmung des Kostenträgers. Zugriff auf diese Datenbank hat nur der Administrator, um im Missbrauchs- oder Fehlerfall die Kontrolle zu behalten. Ein möglicher Log-Eintrag wäre z.B:

log_id	**log_name**	**log_begin**	**log_end**	**log_duration**
34	Erwin Müller	1168018320	1175790720	700
47	Peter Maus	1172018320	1185163072	1243
log_ip	**log_mac**	**log_mobilenr**	**log_hostid**	
85.175.28.33	00:10:DC:45:8D:09	+491705334534	2	
141.61.188.13	AC:04:CC:13:BD:03	+491609824237	1	

Tabelle 6.1: Auszug aus der SpoGA-log-Tabelle

Dies stellt die Felder der Log-Tabelle dar. Eine detailliertere Beschreibung der einzelnen Felder dieser Tabelle folgt in Kapitel 8.4.

7 Software

Neben den bereits installierten Linux-Paketen kommen einige selbstgeschriebene Programme und Skripte zum Einsatz. Dieses Kapitel soll einen Überblick über deren Funktionsweise und auszugsweise auch deren Inhalt geben. Die vollständigen Skripte und Programme sind auf der beiliegenden CD zu finden. In Anhang E ist die Verzeichnisstruktur der Dateien des SpoGA-Systems dargestellt.

7.1 Die Dateien der Web-Oberfläche

Die Web-Oberfläche wurde aus einer Kombination der Skript-Sprache PHP, Java-Skript und HTML entwickelt. Einige Auszüge aus wichtigen Dateien sollen nun vorgestellt werden:

- **index.php**
  Startseite der SpoGA-Oberfläche. Von hier aus kann der Benutzer zu den gewünschten Seiten navigieren und sich am System anmelden, abmelden, seinen Freischalt-Code anfordern, Gastgeber hinzufügen (als Administrator) oder Gastzugänge anlegen (als Gastgeber). Ist ein Gast angemeldet, wird sein derzeitiger Status angezeigt. Dazu wird folgendes PHP-Skript benutzt, das die Daten aus der SQL-Datenbank ausliest und wiedergibt:

```
<?php
/*
 *   Skript zum Auslesen der UserDaten aus der SQL Tabelle
 */
include 'includes/config.inc.php';
// Config-Datei einlesen
$ipadresse = $_SERVER['REMOTE_ADDR'];
// IP-Adresse auslesen
$link = mysql_connect($mysql_host, $mysql_user, $mysql_passwd);
// Verbindung zu MySQL-DB herstellen
if (!$link) {
// Fehlerbehandlung bei  MySQL-Verbindungserstellung
  die('Keine Verbindung moeglich: ' . mysql_error());
}
```

  Allgemeine Anmerkung: „/*...*/“ sowie „//“ dient in PHP zur Kommentierung des Codes.

  Hier wird zunächst die Konfigurationsdatei eingebunden, wo unter anderem auch die Verbindungsdaten für die MySQL-Datenbank festgelegt werden. Diese muss durch einen Administrator bei Installation des SpoGA-Systems einmalig angepasst werden. Anschließend wird die aktuelle IP-Adresse des Nutzers zu Kontrollzwecken eingele-

sen und die Verbindung zur Datenbank hergestellt. Ist keine Verbindung möglich, so wird dies per Fehlermeldung auf der Seite angezeigt.

```
else
{
  mysql_select_db($mysql_db) or die("Auswahl der Datenbank
      fehlgeschlagen!");
  // Datenbank auswaehlen
  // Anfragestring: Auslesen aller Daten mit passender IP
 $query = "SELECT *
           FROM jobs
      WHERE (ip='".$ipadresse."')";
  $result=mysql_query($query);
  // Anfrage stellen
```

Nun wird die benötigte Datenbank ausgewählt und auch hier ein fehlerhafter Verbindungsaufbau mit einer Fehlermeldung quittiert. Dann erfolgt die erste SQL-Anfrage, wobei alle Daten (*) der Tabelle *jobs* ausgelesen werden, wo das Feld „ip“ mit der zuvor ausgelesenen IP-Adresse des Nutzers übereinstimmt. Ist also ein Nutzer angemeldet, so existiert auch ein Eintrag in der jobs-Tabelle mit dessen aktueller IP.

```
 if ($daten=mysql_fetch_array($result, MYSQL_ASSOC))
 {
   if ($daten['login']==1)
  {
      echo "Ihre Handynr: ".$daten['mobilenr']."<br>";
      echo "Ihre verbleibende Nutzungszeit: ".$daten['credits
          ']." Sekunden <br>";
      echo "Ihre Nutzungszeit der aktuellen Session: ".$daten['
          duration']." Sekunden <br>";
      echo "Ihre gesamte bisherige Nutzungszeit: ".$daten['
          surftime']." Sekunden <br>";
      echo "Kostenträger: ".$daten['whopays']."<br>(0=kostenlos
          , 1=Sie selbst, 2=Event-Creator, 9=SpoGA Admin/
          Gastgeber)<br>";
      echo "Ihre IP: ".$daten['ip']."<br>";
  }
  else echo "<b>Sie sind nicht eingeloggt</b><br>";
  }
 else echo "<b>Sie sind nicht eingeloggt</b><br>";
 //Datensatz existiert nicht
  mysql_close($link);
  //Verbindung zur DB beenden
}
?>
```

Ist der aktuelle Benutzer zum Zeitpunkt der Abfrage angemeldet, so ist der Wert von „login“ auf 1 gesetzt. In diesem Falle werden seine Daten aus der Datenbank ausgelesen und entsprechend auf der Startseite angezeigt. Anderenfalls erfolgt die Meldung „Sie sind nicht eingeloggt“. Schließlich wird jetzt noch die Verbindung zur Datenbank getrennt, da diese nun nicht mehr benötigt wird.

Bei jedem Aufruf der *index.php* wird dieses Skript erneut ausgeführt, um den aktuellen Status des Nutzers zu überprüfen und anzuzeigen.

- **login.php**
  Formular, zum Anmelden am System. Ist die Nutzungsdauer abgelaufen erfolgt ein automatisches Abmelden und der Benutzer bekommt eine entsprechende Fehlermeldung.

  Auch hier sollen auszugsweise die wichtigen Code-Segmente erläutert werden:

```
/*
 * Java-Skript zum Ueberpruefen der Eingaben
 */
function chkFormular()
{
 if(document.Formular.code.value == "")  {
   alert("Bitte Freischalt-Code eingeben!");
   document.Formular.code.focus();
   return false;
  }
}
```

  Dieses Java-Skript dient zum Überprüfen der Eingabe. Falls kein Code angegeben wurde, erscheint ein Warn-Fenster „Bitte Freischalt-Code eingeben!“. Ein ähnlich aufgebautes Skript kommt bei den anderen Programmdateien häufiger zum Einsatz und wird daher an dieser Stelle explizit erwähnt.

```
if (!isset($_POST['code'])) {}
else
{
  $query = "SELECT *
            FROM jobs
            WHERE code='".$_POST['code']."'";
  $result=mysql_query($query);
  // SQL-Anfrage stellen
  if ($data=mysql_fetch_array($result, MYSQL_ASSOC))
  // Array der Daten erstellen
  {
    echo "Ihre Handynr: ".$data['mobilenr']."<br>
          Restnutzungzeit: ".$data['credits']."<br>
          Kostenträger: ".$data['whopays']."<br>
          ";
```

  Danach erfolgt wieder ein Verbindungsaufbau zur Datenbank (siehe bei index.php) und eine SQL-Anfrage, diesmal abhängig vom eingegebenen Code. `$_POST['code']` liest dabei den Code aus, der auf dieser Seite im Feld Code eingegeben wurde. Dieser wird nun in der jobs-Tabelle gesucht und die dazugehörende Daten (Handynummer, Restnutzungszeit, Kostenträger) angezeigt.

  Daraufhin folgt ein Code-Segment, das etwas komplexer ist und an dieser Stelle nicht näher erläutert wird. Dabei wird überprüft, ob noch Restnutzungszeit verfüg-

bar ist und wenn ja, ob eine aktuelle Veranstaltung zum eingegeben Code existiert. Ist auch dies gewährleistet, so wird der Nutzer an der Firewall freigeschaltet:

```
// iptables zum Freischalten der Nutzung ausfuehren
exec('sudo iptables -I FORWARD -s '.$ipadresse.' -i eth1 -j
    ACCEPT');
exec('sudo iptables -I FORWARD -d '.$ipadresse.' -i eth0 -j
    ACCEPT');
exec('sudo iptables -I PREROUTING -t nat -p tcp -s '.$ipadresse
    .' --dport 80 -j ACCEPT');
```

Dabei werden alle Daten, die über Port 80 (HTTP) von der IP-Adresse des Nutzers kommen, weitergeleitet zur 2. Netzwerkkarte, an dem das Firmennetz angeschlossen ist. Der Nutzer hat jetzt also Zugang zum Firmennetz und hierfür liegt der Verantwortungsbereich nun bei der Firma selbst.

```
$query1 = "INSERT INTO log
           (manager_id,log_name,log_mobilenr,log_ip,log_begin,
               log_whopays,log_mac)
           VALUES ('".$data['manager_id']."','".$data['name
               ']."','".$data['mobilenr']."',
           '".$ipadresse."','".$beginArray[0]."','".$data['
               whopays']."','".$mac."')";
$query2 = "UPDATE jobs
           SET login=1,duration=0,ip='".$ipadresse."',mac='".$
               mac."'
           WHERE ((mobilenr='".$data['mobilenr']."') AND (code
               ='".$_POST['code']."'))";
      $result1=mysql_query($query1);     //SQL-Anfrage stellen
      $result2=mysql_query($query2);     //SQL-Anfrage stellen
```

Anschließend werden 2 SQL-Anfragen ausgeführt. Mit query1 erfolgt ein Eintrag in der log-Tabelle, wo die Verbindungsdaten des Nutzers samt Start-Zeit festgehalten werden. Durch query2 wird sichergestellt, das sich kein weiterer Nutzer mit identischem Code anmelden kann.

- **logout.php**
  Formular, zum (manuellen) Abmelden vom System. Dabei werden nach Eingabe des aktuell gültigen Codes, mit dem ein Benutzer angemeldet ist, die Firewall-Regeln, die durch *login.php* eingetragen wurden, wieder gelöscht (iptables -D ..., -D steht für delete):

```
exec('sudo iptables -D FORWARD -s '.$ipadresse.' -i eth1 -j
    ACCEPT'); //Sperren des Rechners
exec('sudo iptables -D FORWARD -d '.$ipadresse.' -i eth0 -j
    ACCEPT');
exec('sudo iptables -D PREROUTING -t nat -p tcp -s '.$ipadresse
    .' --dport 80 -j ACCEPT');
```

- **getcode.php**
  Durch Angabe seiner Handy-Nummer kann ein Benutzer hier seinen Freischalt-Code zum Anmelden am System erneut anfordern, sofern dieser aktuell und gültig ist.

Nach Validierung des Codes wird dann das Skript zum Senden der SMS aufgerufen und der Code erneut an die dazugehörige Handy-Nummer geschickt. Für Details hierzu siehe Kapitel 7.2.

- **addhost.php**
  Mit einem gültigen Administrator-Zugang können hier Gastgeber angelegt werden. Erforderliche Angaben dabei sind der Name des Gastgebers, sein späteres Passwort und seine E-Mail-Adresse, um ihn bei Rückfragen kontaktieren zu können. Diese Angaben werden nun in die hosts-Tabelle der Datenbank geschrieben:

```
$query = "INSERT INTO hosts (host_name,host_pass,host_email,
    admin_id)
      VALUES('".$_POST['name']."','".$_POST['pass']."',
      '".$_POST['email']."','".$_POST['adminid']."')";
```

  Der Gastgeber-Zugang ist nun aktiv und kann der entsprechenden Person übermittelt werden.

- **addguest.php**
  Dies ist das bereits in Abbildung 6.4 vorgestellte Formular zum Einrichten von Gastzugängen. Um Gäste anlegen zu können, ist ein Gastgeber-Zugang nötig. Durch Angabe von Name und Handy-Nummer des Gastes sowie des gewünschten Startzeitpunktes mit entsprechender Nutzungszeit in Minuten wird anschließend ein Eintrag in der „ jobs“-Tabelle vorgenommen. Nun wird eine Stunde vor dem gewählten Startzeitpunkt, ab dem der Gastzugang gültig ist, der nötige Freischalt-Code per SMS an den Gast gesendet. Dieser kann sich dann am System anmelden, bis seine Nutzungsdauer abgelaufen ist.

  Der zugehörige Source-Code soll an dieser Stelle nicht explizit erläutert werden, da er den bereits vorgestellen Listings ähnelt und lediglich in der Struktur der SQL-Anfragen unterscheidet.

- **deleteguest.php**
  Werden Gäste hinzugefügt, so muss natürlich auch die Möglichkeit bestehen, diese wieder zu löschen. Dies wird mittels der Datei deleteguest.php realisiert. Dabei wird eine Liste der Teilnehmer erstellt und nach Eingabe der ID des zu löschenden Eintrages wird dieser aus der Datenbank entfernt:

```
$query2 = "SELECT * FROM jobs WHERE (job_id='".$_POST['deleteid
    ']."')";
$query3 = "DELETE FROM jobs WHERE (job_id='".$_POST['deleteid
    ']."')";
$result2=mysql_query($query2);      // SQL-Anfrage stellen
$result3=mysql_query($query3);      // SQL-Anfrage stellen
if ($data=mysql_fetch_array($result2, MYSQL_ASSOC))
{
 if ($result3) print "<b>Datensatz erfolgreich gelöscht!</b><br
     >";
 else print "<b>Datensatz nicht gelöscht!</b>";
}
else
```

```
{
 print mysql_error();
 print "<br>";
 print "<b>Datensatz existiert nicht! Bitte gültige ID eingeben
     !</b>";
}
```

Diese Funktion erlaubt es einem Administrator alle Gäste des Systems zu löschen, und einem Gastgeber die von ihm angelegten Zugänge.

- **deletehost.php**
  Ebenso wie die Gastzugänge können auch die Gastgeberzugänge wieder entfernt werden. Dies kann nur durch einen Administrator erfolgen. Es wird ebenfalls eine Liste der vorhandenen Gastgeber erzeugt, die dann einzeln gelöscht werden können.

- **list-guests.php**
  Um eine Übersicht der eingetragenen Gäste im System zu erhalten, wird hiermit eine Liste aller Gäste erstellt. Dazu wird eine Administrator-Zugang benötigt.

```
if ($_POST['adminid'] == $admin_id and $_POST['adminpass'] == $
    admin_passwd)
// überprüfen des Admin-Zugangs
{
  $query = "SELECT * FROM jobs ORDER BY job_id";
 $result=mysql_query($query);      // SQL-Anfrage stellen
 if ($result)
  {
    print "<table border=\"1\" cellspacing=\"0\" ".
   "cellpadding=\"5\" WIDTH=\"100\%\" >\n";
   print "<tr><th>ID</th><th>Name</th><th>Handynummer</th><th>
       Startzeitpunkt</th>
   <th>Restguthaben</th><th>Gastgeber-ID</th><th>eingeloggt?</
       th>\n";
   while ($dataset = mysql_fetch_array($result))
     {
      $datum = date("d.m.Y H:i",$dataset['begin']);
      print "<tr>";
     print "<td>".$dataset['job_id']."</td>";
     print "<td>".$dataset['name']."</td>";
     print "<td>".$dataset['mobilenr']."</td>";
     print "<td>".$datum."</td>";
     print "<td>".$dataset['credits']."</td>";
     print "<td>".$dataset['host_id']."</td>";
     print "<td>".$dataset['login']." "."</td>";
     print "</tr>\n";
    }
    $count = mysql_num_rows($result);
    print "<tr><td colspan = \"9\">";
    print "Es wurden $count Datensätze gefunden. Die Liste ist
        sortiert nach ID.";
    print "</td></tr>\n";
    print "</table>\n";
  }
```

```
 else echo "<b>Auslesen fehlgeschlagen oder kein Datensatz
     vorhanden!</b><br>";
 mysql_close($link);    //Verbindung zur DB beenden
}
```

Die Daten der Gäste werden aus der Datenbank ausgelesen und in Tabellenform angezeigt (siehe Abb. 6.5). Das Listing zeigt ein Auszug dieser Datei.

Die gleiche Funktion steht auch für Gastgeber zur Verfügung. Nach dessen Anmelden kann er sich hierbei alle Gäste, die von ihm angelegt wurden, anzeigen lassen. Dabei wird auch der Freischalt-Code, den die Gäste zum Anmelden benötigen angezeigt und der Gastgeber kann dem Gast somit bei möglichen Anmelde-Problemen schnell weiterhelfen.

- **list-hosts.php**
  Vergleichbar mit *list-guests.php*, allerdings mit der Einschränkung, dass nur Administratoren diese Funktion benutzen können. Hier werden die von einem Administrator angelegten Gastgeber-Zugänge aufgelistet.

- **header.php + left.inc.php**
  Dateien, die für die Formatierung und Menüstruktur der Web-Oberfläche nötig sind.

- **Verzeichnis „includes“**
  Hier befindet sich die Konfigurations-Datei, wo die individuellen Zugangsdaten zur MySQL-Datenbank und zum SMS-Gateway festgelegt werden. Bei Bedarf können diese geändert bzw. angepasst werden. Die css-Datei dient zum Anpassen des Erscheinungsbildes der Web-Oberfläche (Farben, Schriftarten, Schriftgrößen etc.).

- **Verzeichnis „images“**
  Nötige Hintergrundbilder, Logos oder sonstige Bilddateien der Web-Oberfläche

7.2 Dateien für den SMS-Versand

- **sendsms.php**
  Mittels dieser Datei werden die SMS vom Server versendet. Der Inhalt ist Abhängig vom jeweiligen SMS-Gateway-Dienstleister (hier wird der Gateway von sms77.de[1] verwendet). Um SMS versenden zu können, wird ein gültiges Benutzerkonto vorausgesetzt. Preise und Konditionen sind auf den entsprechenden Anbieter-Seiten nachzulesen.

```
<?PHP
/*
  * Script zum Senden von SMS ueber SMS-Gateway (hier SMS77.de)
  * config-Datei mit weiteren Variablen wird benoetigt
  */
function send($sms_to,$sms_msg)
{
 include("includes/config.inc.php");
```

[1] http://www.sms77.de/

```
 // Konfigurationsdatei einbinden
  $msg = urlencode($sms_msg);
 // Textnachricht wird für URL-Übergabe codiert
 $url = 'http://sms77.de/gateway/?u='.$sms_user.'&p='.$sms_
     passwd.'&to='.$sms_to.'&text='.
 $msg.'&type=basicplus&from='.$sms_from.'';
 $ret = @file($url);
 // hier erfolgt der Aufruf des HTTP-APIs mittels http-Request
 // das "'@"' ist erforderlich, damit die URL bei Fehler nicht
     mit ausgegeben wird
 if ($ret[0] == "100")
 {
  echo "SMS erfolgreich versendet!\n";
  return true;
 }
 else
 {
  echo "Fehler beim SMS-Versand! Fehlercode: ".$ret[0]."\n";
  // Fehlercodeausgabe
  return false;
 }
}
?>
```

Zunächst wird wieder die Konfigurations-Datei eingebunden, in der Benutzername und Password des SMS-Gateways stehen. Nun wird die Funktion „send“ mit den nötigen Parametern aufgerufen. `sms_to` ist dabei die Handynummer, wohin die Nachricht gesendet werden soll, `sms_msg` ist die zu sendende Nachricht an sich. Der verwendete Gateway-Anbieter SMS77 verlangt eine URL-codierte Nachricht, die mit der PHP-Funktion „urlencode(Text...)“ erzeugt wird und weitere Optionen, die in der Dokumentation der SMS HTTP-API (siehe Anhang C) nachzulesen sind. Im Falle eines Fehlers erfolgt eine Fehlermeldung mit zugehörigem Fehlercode (Ebenfalls in der Dokumentation nachzulesen).

- **sms-wrapper.php**
  Wrapper-Skript das benötigt wird, um die *sendsms.php* aus dem C++ Programm heraus aufzurufen. Dabei werden lediglich 2 Konsolen-Argumente an die send-Funktion aus der *sendsms.php* übergeben.

```
<?php
/*
 * Wrapper-Script um send() aus dem C++ Programm heraus
     auzufuehren
 * command: php -f sms_wrapper 00491751111111 'Text der SMS...'
 * Format der Handynr ist 0049170.... oder 49170.... oder
     0170...
 */
require_once 'sendsms.php';
echo send($argv[1],$argv[2]);
?>
```

7.3 Dateien für den Webservice

Die Basis des Webservices bildet ein NuSOAP-Server. Dieser stellt Funktionen bereit, die ein Client mittels SOAP-Protokoll abrufen und nutzen kann. Weitere Informationen zu NuSOAP findet man auf der offiziellen NuSOAP-Seite[2].

- **server.php**
  Webservice-Server, der Funktionen bereitstellt zum Senden von Daten zum E-IMS-Server, Senden von SMS vom E-IMS-Server aus, und zum Abrufen von Informationen für die Rechnungsstelle. Als Webservice-Client dient in diesem Falle der E-IMS-Server.

- **config.php**
  Konfigurations-Datei, in der die individuellen Zugangsdaten zur MySQL-Datenbank, zum SMS-Gateway sowie zur Authentifikation am SpoGA-System mittels Passwort festgelegt werden. Bei Bedarf bitte ändern bzw. anpassen.

- **sendsms.php**
  Datei zum SMS-Versand. Beschreibung siehe oben.

7.4 Das C++-Programm

- **spoga**
  Programm zur Überwachung des SpoGA Servers. Wenn ein User eingeloggt ist, werden seine Nutzungszeit und die aktuelle Sitzungsdauer überwacht. Falls die Nutzungszeit abgelaufen, oder eine Veranstaltung ausgelaufen ist, wird die IP des Nutzers von der Firewall wieder geblockt. Eine Stunde vor Veranstaltungsbeginn bekommt der Benutzer eine SMS mit dem Freischalt-Code zu seinem aktuellen Event gesendet.

  Dieses Programm befindet sich auf dem SpoGA-Server selbst und läuft ständig im Hintergrund. Wird es beendet, so sind die Überwachungsfunktionen sowie die Protokollierung nicht mehr aktiv, die Zugänge aber noch funktionsfähig.

- **spoga.cpp**
  Source-Code des Programms.

7.5 Das Firewall-Skript

- **firewall**
  Auf dem Skeleton-Skript aus /etc/init.d/ basierendes Skript, welches die Firewall-Regeln beinhaltet. Es werden zunächst alle existierenden Regeln gelöscht, um eventuell existierende Firewalls auszuschalten. Alle ankommenden Pakete werden gedroppt, also verworfen und es werden nur HTTP-Anfragen (Port 80) und DNS-Anfragen (Port 53) durchgelassen. Zusätzlich wird man automatisch auf die Startseite des Webservers geleitet.

[2]NuSOAP-Seite: http://dietrich.ganx4.com/index.php?id=132

8 Die Datenbank

8.1 Die Datenbank-Struktur

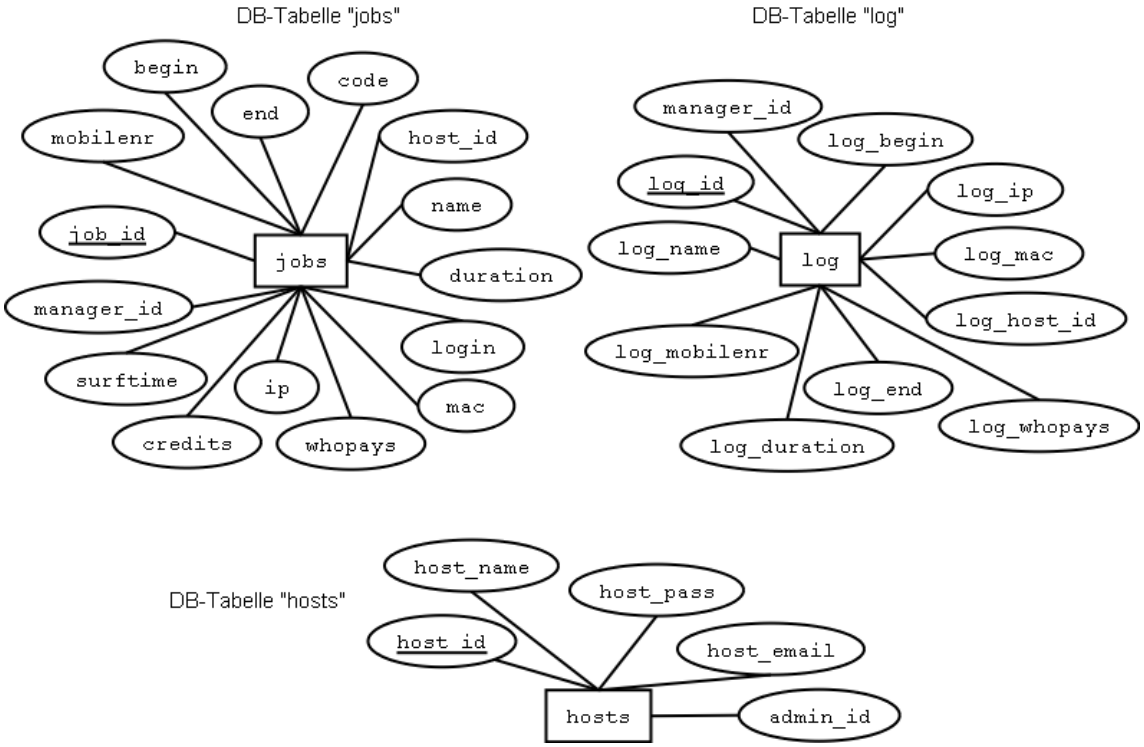


Abbildung 8.1: Datenbank-Struktur

Eine zentrale Rolle des SpoGA-Servers spielt die Datenbank. Hier werden die Benutzerdaten, Gastgeberdaten und Logs abgespeichert. Es gibt zunächst eine Datenbank namens „spoga“. Diese enthält drei Tabellen „ jobs“, „hosts“ und „log“ (siehe Abb. 8.1). Die jeweiligen Felder werden in diesem Abschnitt etwas detaillierter beschrieben, um die Relevanz der SpoGA-Datenbank zu verdeutlichen

8.2 Struktur der jobs-Tabelle

Diese Tabelle enthält die eigentlichen Daten der Gäste.

- **job_id**
  Primärschlüssel dieser Tabelle. Fortlaufende Nummer zur eindeutigen Identifizierung eines Eintrages.
- **mobilenr**
  Handynummer des Teilnehmers/Gastes

- **begin**
  Beginn einer Veranstaltung im Unix-Timestamp-Format[1].
- **end**
  Ende einer Veranstaltung im Unix-Timestamp-Format.
- **code**
  8-stelliger Code zum Anmelden am System.
- **manager_id**
  Identifikations-Schlüssel für E-IMS-System
- **surftime**
  Nutzungszeit der aktuellen Sitzung in Sekunden.
- **credits**
  Verfügbare Rest-Nutzungszeit in Sekunden.
- **whopays**
  Kostenträger: Gibt an, wer für die enstehenden Online-Kosten aufkommen soll:
  - 0: kostenfreie Nutzung
  - 1: Der Gast trägt selbst die Kosten, die er verursacht
  - 2: Der Gastgeber trägt die Kosten, die durch Gäste verursacht werden
- **ip**
  IP-Adresse des Gastes/Teilnehmers
- **mac**
  MAC-Adresse des Gastes/Teilnehmers
- **login**
  Flag zum Status der Anmeldung:
  - 0: Die Person ist nicht angemeldet
  - 1: Die Person ist angemeldet
- **duration**
  Zeit der Sitzungs-Dauer nach Abmelden für eventuelle Auswertungen in Sekunden (end - begin)
- **host_id**
  ID des Veranstalters bzw. des Gastgebers.
- **host_pass**
  Passwort des Veranstalters bzw. des Gastgebers.

[1] Unix-Timestamp: http://de.wikipedia.org/wiki/Timestamp

8.3 Struktur der hosts-Tabelle

In dieser Tabelle sind die Daten der Gastgeber abgelegt.

- **host_id**
  ID des Veranstalters bzw. des Gastgebers.
- **host_name**
  Names des Veranstalters bzw. des Gastgebers.
- **host_pass**
  Passwort des Veranstalters bzw. des Gastgebers.
- **host_email**
  E-Mail des Veranstalters bzw. des Gastgebers.
- **admin_id**
  ID des Administrators, der den Gastgeber hinzugefügt hat.

8.4 Struktur der log-Tabelle

Tabelle mit Log-Einträgen.

- **log_id**
  Primärschlüssel dieser Tabelle. Fortlaufende Nummer zur eindeutigen Identifizierung eines Eintrages.
- **manager_id**
  Identifikations-Schlüssel für E-IMS-System
- **log_begin**
  Beginn des Log-Eintrages im Unix-Timestamp-Format (Anmelden am System)
- **log_end**
  Ende des Log-Eintrages im Unix-Timestamp-Format (Abmelden vom System)
- **log_ip**
  IP-Adresse des Gastes/Teilnehmers
- **log_mac**
  MAC-Adresse des Gastes/Teilnehmers
- **log_duration**
  Zeit der Sitzungs-Dauer nach Abmelden für eventuelle Auswertungen in Sekunden (end - begin)
- **log_mobilenr**
  Handynummer des Teilnehmers/Gastes
- **log_whopays**
  Kostenträger: Gibt an, wer für die enstehenden Online-Kosten aufkommen soll:

  - 0: kostenfreie Nutzung
  - 1: Der Gast trägt selbst die Kosten, die er verursacht
  - 2: Der Gastgeber trägt die Kosten, die durch Gäste verursacht werden

- **log_host_id**
  ID des Veranstalters bzw. des Gastgebers bei manueller Freischaltung der Gäste.

9 Die Webservice-Schnittstelle

Da die Webservice-Schnittstelle die Verbindung von SpoGA- und E-IMS-System darstellt und somit eine wichige Rolle im Bezug auf das Gesamt-Projekt einnimmt, soll sie in diesem Kapitel etwas detaillierter erläutert werden. Zunächst wird einleitend auf die Grundlagen eines Webservices eingegangen. Dann folgt eine Auflistung der wichtigsten Funktionen, die in dieser Arbeit mit Hilfe von Webservices umgesetzt wurden. Abschließend wird noch kurz der Quelltext dieser Funktionen exemplarisch beschrieben.

9.1 Grundlagen

Webservices wurden als serviceorientierte Architekturen entwickelt, um entfernte Funktionsaufrufe über ein Kommunikationsmedium (im Wesentlichen das Internet) ermöglichen zu können. Dabei wurden sie von Anfang an so konzipiert, dass sie prinzipiell Plattformunabhängigkeit bieten. Sie reduzieren die Komplexität der Verbindungen zwischen verteilten Systemen, indem ein einheitliches Interface für Funktionsaufrufe bereitgestellt wird. Führende Softwareunternehmen, u.a. Microsoft und IBM, setzen sich intensiv für Webservices ein. Die Verwendung offener Standards und die Unabhängigkeit von Programmiersprachen sprechen für den Einsatz von Webservices (vgl. [CW03] Kapitel 1).

Endanwender benutzen Webservices unbewusst, da die dahinter steckende Logik verborgen bleibt. Kunden können z.B. in einem Online-Shop jederzeit den Lieferstatus ihrer Bestellung erfragen. Dass die gewünschten Informationen nicht vom Betreiber selbst kommen, sondern möglicherweise von dem Logistikdienstleister geliefert werden, nimmt der Kunde dabei nicht wahr. In diesem Fall wären der Online-Shop-Betreiber und dessen Logistikpartner über Webservices zu einem E-Business-System miteinander verbunden.

Web Services bauen im Wesentlichen auf XML (Extensible Markup Language) in Verbindung mit Internet-Protokollen wie HTTP oder SMTP auf (vgl. [Man06]). Als großen Vorteil der Webservice-Technologie erweist sich ein standardisierter Kommunikationsmechanismus zwischen verteilten Anwendungen. Webservices können mit einer großen Interoperabilität und Erweiterbarkeit aufwarten, da sie leicht miteinander kombinierbar sind.

Die Grundbausteine von Webservices stellen die Protokolle WSDL (Web Services Description Language), SOAP (Simple Object Access Protocol) und UDDI (Universal Definition, Discovery and Integration) dar.

- **WSDL** beschreibt die Schnittstellendefinitionen von Webservices. Mit Hilfe von XML beschreibt WSDL die Art der Funktionalität und wie das Format der Ein- und Ausgabenachrichten aussieht.
- **SOAP** ist das Kommunikationsprotokoll zwischen Webservice-Anwendungen. SOAP definiert Regeln für die Architektur, den Transport und den einfachen Austausch von XML-Dokumenten zwischen verteilten Systemen.

- **UDDI** ist eine standardisierte Beschreibungssprache, die es einem Webservice-Anbieter ermöglicht, seine Services in einer sogenannten *UDDI Business Registry*[1] zu veröffentlichen.

In Dienstkatalogen werden Informationen über Webservices angeboten. Verzeichnisse stellen Leistungs- und Adressdaten (Wo befindet sich der Webservice? Welche Struktur hat er?) sowie Informationen zu den verfügbaren Schnittstellen bereit. UDDI-Repositories werden dazu verwendet, um Webservices zu verwalten und auf sie zuzugreifen. Das Zusammenspiel zwischen UDDI, WSDL und SOAP ist in Abb. 9.1 dargestellt.

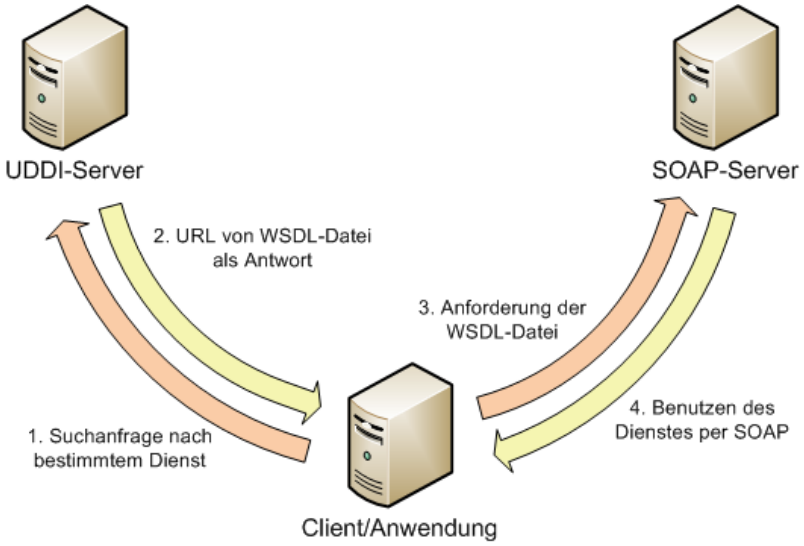


Abbildung 9.1: Zusammenspiel UDDI, WSDL und SOAP (Quelle: [GA04])

Aufgrund des Servicegedankens können Webservices sehr leicht miteinander kombiniert werden, um komplexere Prozesse mit Nutzen für ein Unternehmen zu realisieren. Außerdem nimmt die Wiederverwendbarkeit zu. Diese Features sprechen für den Einsatz von Webservices in einem Integrationsprozess verschiedener Informationssysteme. Webservices sind modulare Geschäftsanwendungen, die weborientiert sind, Vorteile eines offenen Standards bieten und somit eine vereinfachte Erstellung von Schnittstellen ermöglichen (vergleiche dazu [GA04]).

[1] UDDI Business Registry: `http://www.uddicentral.com/`

9.2 Einsatz von Webservices im SpoGA-System

Die Webservice-Schnittstelle dieses Projektes bietet 5 grundlegende Funktionen, die eine Kommunikation zwischen SpoGA- und E-IMS-System gewährleistet. Hierbei handelt es sich um:

- das Testen der Kommunikation zwischen E-IMS und SpoGA sowie dessen Konfiguration
- das Eintragen von Benutzer-Daten in der SpoGA-Datenbank
- die Änderung der Mobilfunknummer eines bestehenden Benutzers
- das Versenden des Freischaltcodes per SMS an den Benutzer
- das Senden der Abrechnungsdaten eines Benutzers an E-IMS

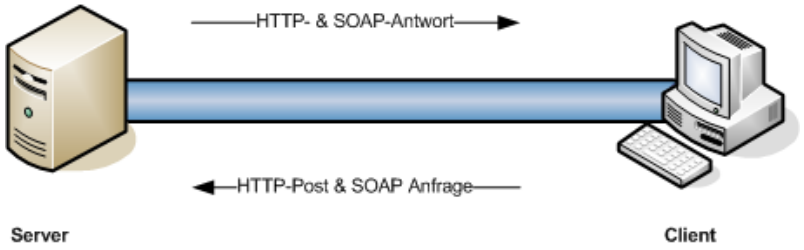


Abbildung 9.2: Die Webservice-Kommunikation

In dieser Arbeit wird der SOAP-Standard ohne UDDI und WSDL als Basis der Webservice-Schnittstelle verwendet, da dies für den Prototyp zunächst ausreichend ist. Abb. 9.2 verdeutlicht die Kommunikation zwischen einem Server und einem Client. Das SpoGA-System stellt in diesem Fall den Webservice-Server dar, das E-IMS-System den Webservice-Client. Mit anderen Worten bedeutet das: der SpoGA-Server stellt die o.g. Funktionen bereit, die der E-IMS-Server dann nutzen kann. Dazu sendet der Client eine SOAP-Anfrage mittels HTTP-Request an den Server und dieser antwortet mit einer entsprechenden SOAP-Antwort. Die Kommunikation ist dabei einseitig. Der SpoGA-Server stellt alle Webservices zur Verfügung und hat keine Möglichkeit von sich aus den E-IMS Server zu kontaktieren.

Ein kleines Szenario zur Verdeutlichung:
Die Mobilfunknummer von Horst Mayer hat sich geändert. Er nutzt SpoGA in Kombination mit E-IMS und ist somit in der Groupware eingetragen. In Kürze wird er an einer Tagung teilnehmen und sein Gast-Zugang ist bereits in der SpoGA-Datenbank angelegt worden. Nun kann das E-IMS-System die Mobilfunknummer von Herrn Mayer dort ändern, indem es die Webservice-Funktion `update\_number nutzt`. Diese durchsucht die SpoGA-Datenbank nach der ID von Kunde Mayer und aktualisiert dessen Mobilfunknummer. Die erfolgreiche Änderung wird durch eine „OK“-Antwort bestätigt und der Vorgang ist abgeschlossen.

9.3 Der Quellcode

Um die erstellten Funktionen zu beschreiben, wird im folgenden Abschnitt exemplarisch ein Teil des entwickelten Quellcodes erläutert. Zunächst ein Ausschnitt der Datei *server.php*.

```
<?PHP
$ipadresse = $_SERVER['REMOTE_ADDR'];

require_once("nusoap.php");

$s=new soap_server;

$s->register('senddatabe');
// Events von Groupware in Job-Tabelle eintragen
$s->register('com_test');        // Kommunikationstest
$s->register('send_sms');        // SMS-Sendefunktion
$s->register('send_bill');       // Daten fuers billing
$s->register('update_number');   // zum updaten der Handynummer
```

Nach Auslesen der IP-Adresse des Clients wird die wichtigste Datei des Webservices, *nusoap.php*, eingebunden. Sie wird benötigt, um SOAP über PHP nutzen zu können. Diese wurde auf der offiziellen NuSOAP-Seite[2] heruntergeladen und in das PHP5-Include-Verzeichnis (/usr/share/php5/) abgelegt. Alternativ kann man die Datei auch in ein anderes Verzeichnis ablegen, dann muss man den kompletten Pfad allerdings explizit angeben. Eine detailierte Installationsbeschreibung von NuSOAP ist in [CW03] Kapitel 3 nachzulesen.

Mit dem Befehl `\$s=new soap_server` wird ein neuer SOAP-Server erzeugt und der Variable `s` zugewiesen. Für diesen SOAP-Server müssen jetzt noch die gewünschten Funktionen registriert werden, um sie auch von einem Client abrufbar zu machen. Eine solche Registrierung erfolgt mittels des PHP-Befehls `register('...')`. Hiermit werden die einzelnen Funktionen dem SOAP-Server zugeodnet und sind somit über diesen ansprechbar.

Die Funktion *senddatabe(...)* dient dazu, die „Jobs“ der Groupware in die Datenbank des SpoGA-Servers einzutragen. Dazu werden Daten wie Handynummer des Nutzers, Veranstaltungsbeginn und -endzeit, die Nutzer-ID, und Kostenträger-Informationen als Parameter übergeben.

```
function senddatabe($handynr,$begin,$end,$managerid,$server_passwd
    ,$checksum,$duration,$whopays) {
```

Besonders zu erwähnen sind dabei die Parameter *server_passwd* und *checksum*. Diese dienen als „Sicherheits-Prüfung“ des entwickelten Webservices. Es soll somit verhindert werden, dass ein Angreifer sich als Webservice-Client ausgibt und so die vertraulichen Nutzerdaten der Groupware abfragen kann.

Im folgenden Code-Stück wird die genaue Zusammensetzung dieser Parameter etwas deutlicher:

[2] NuSOAP:http://dietrich.ganx4.com/nusoap/

```
include_once("config.php");
if ($server_passwd == md5($serverpasswd))
{
  $sum = $handynr[1]+$handynr[2]+$handynr[4]+$handynr[5]+$handynr
      [6]+$handynr[8]+$handynr[9];
  $tmp = ($sum+$begin+$end+$duration+$whopays);
  $testsum = md5($tmp);
  if ($testsum == $checksum)
  {
    $link = mysql_connect($mysql_host, $mysql_user, $mysql_passwd);
    ...
```

Zunächst wir die Datei *config.php* eingebunden, die unter anderem auch das spezifische SpoGA-Serverpasswort (serverpasswd) enthält. Dieses wird vom Webservice-Client (hier der E-IMS-Server) in MD5-codierter Form[3] übergeben und mit dem lokalen Serverpasswort aus der Konfigurationsdatei verglichen. Stimmen die Passwörter überein, wird eine Prüfsumme (*testsum*) generiert:

1. einzelne Stellen der Handynummer des Nutzers werden aufaddiert zu *sum*, um eine möglichst zufällige Zahl zu generieren
2. es wird der Begin-Zeitpunkt der zugehörigen Veranstaltung im Timestamp-Format dazuaddiert (*begin*)
3. es wird der End-Zeitpunkt der zugehörigen Veranstaltung im Timestamp-Format dazuaddiert (*end*)
4. es wird die Dauer der aktuellen Sitzung in Sekunden dazuaddiert (*duration*)
5. es wird die ID des Kostenträgers dazuaddiert (*whopays*)

Nun wird die so entstandene *testsum* ebenfalls MD5-codiert und mit der *checksum* des Webservice-Clients verglichen.
Sind auch diese identisch (`if ($testsum == $checksum)`), wird erst die Verbindung zur SpoGA-Datenbank aufgebaut. Die Erzeugung der Prüfsumme muss natürlich vom Client äquivalent erfolgen. Dieses selbstentwickelte Verfahren kommt in den anderen Funktionen ebenfalls zum Einsatz, allerdings mit anders generierten Prüfsummen, um die Sicherheit nochmals zu erhöhen. So kann z.B. ein potentieller Angreifer, der die MD5-codierte Prüfsumme abgefangen hat, keine andere Funktion mit dieser Prüfsumme verwenden.

Nach erfolgreich aufgebauter Verbindung zur SpoGA-Datenbank wird nun der Code, der zum Anmelden auf der SpoGA-Startseite benötigt wird und später per SMS an den Nutzer gesendet wird, generiert:

```
...
$dateArray = getdate();
$tmp = $dateArray[0] + 4711 ;
// geheimer Schlüssel für den Code = Aktueller Unix-Timestamp +
    4711
$codetmp = md5($tmp);           // verschlüsselung des Code
```

[3] http://de.wikipedia.org/wiki/Message_Digest_Algorithm_5

```
$code = substr($codetmp, 0, 8); // Nur die ersten 8 Stellen des
    Code
...
```

Der Code setzt sich dabei wie folgt zusammen:

1. das aktuelle Datum im Timestamp-Format wird ausgelesen
2. dieses Datum wird mit einer frei gewählten Zahl addiert (hier 4711)
3. diese Summe (*tmp*) wird nun MD5-codiert (*codetmp*)
4. um den SMS-Code nicht zu lang werden zu lassen werden nur die ersten 8 Stellen benutzt

Nun kann der eigentliche Eintrag in die Datenbank-Tabelle „jobs“ erfolgen:

```
...
$query = "INSERT INTO jobs (mobilenr,begin,end,manager_id,whopays,
    surftime,duration,ip,login,credits,code)
 VALUES('".$handynr."','".$begin."','".$end."','".$managerid
     ."','".$whopays."',0,0,'0.0.0.0',0,'".$credits."','".$code."')
     ";
$result=mysql_query($query);

if (!$result)
{
  return "No Data";
}
else
{
  return "OK";
}
mysql_close($link);
...
```

Dabei werden sämtliche Daten eingetragen, auch der soeben erzeugte Code (*code*) sowie die IP-Adresse des Nutzers. Da diese zum jetzigen Zeitpunkt noch nicht bekannt ist, wird „0.0.0.0“ eingetragen. Sobald der Nutzer sich am System angemeldet hat, wird diese durch durch das Anmelde-Script ermittelt und in der Datenbank aktualisiert. Durch das Senden des Strings „OK“ wird der Vorgang quittiert oder mit „No Data“ als fehlerhaft angezeigt, bevor die Verbindung zur Datenbank wieder getrennt wird.

Wie sieht es mit der Sicherheit dieses selbstentwickelten Verfahrens aus?

MD5 verwendet einen Hashwert von 128 Bit Länge. Dieser Hashwert wird üblicherweise als 32-stellige Hexadezimalzahl notiert, die dann beispielsweise so aussehen kann:
34048ce4cd069b624f6e021ba63ecde5

Eine solche Codierung ist zwar nicht 100 prozentig sicher, da man durch „Ausprobieren“ sämtlicher Kombinationen von Zahlen und Buchstaben theoretisch auf den richtigen Wert kommen kann (vgl. [Sch02] Kapitel 1), es lassen sich aus diesem Wert aber keinerlei Rückschlüsse auf den ursprünglichen Code (vor Anwendung des MD5) ziehen. Hinzu

kommt, dass sich der erzeugte Code, wie oben beschrieben, bei jeder Generierung ändert. Daher bietet dieses Verfahren ausreichende Sicherheit für den Anwendungsfall.

Die übrigen Webservice-Funktionen sind ähnlich dem aufgeführten Beispiel aufgebaut und sollen an dieser Stelle nicht näher erläutert werden. Der interessierte Leser findet in Anhang G.3 den kompletten Quellcode der *server.php*. Ein Beispiel für den client-seitigen Zugriff auf die Webservice-Schnittstelle ist in der Arbeit von Christoph Speich ([Spe07] Kapitel 7) nachzulesen.

10 Anpassung des Systems

Um den SpoGA-Server nutzen zu können, müssen noch einige Änderungen bzw. Anpassungen am Basissystem vorgenommen werden. Diese sollen im folgenden Text kurz erläutert werden.

- Anpassen der Benutzer-Rechte
  Der Apache-Webserver läuft als sogenannter „www-User“. Alle PHP-Skripte oder Kommandos, die über die Webseite aus aufgerufen werden, haben somit aus Sicherheitsgründen eingeschränkte Zugriffsrechte auf das System. Da die Firewall des SpoGA-Systems aber über solche Skripte gesteuert wird (z.B. beim An- bzw. Abmeldevorgang), benötigt der www-User eine Möglichkeit diese zu ändern. Dazu ist der iptables-Befehl gut, welcher normalerweise nur dem root-User vorbehalten ist. Das erforderliche Software-Paket „sudo“ wurde ja, wie oben beschrieben, bereits installiert.

  Nun muss die Datei „sudoers“, die sich im /etc/ Verzeichnis befindet, angepasst und um folgenden Eintrag ergänzt werden:

```
# fuer SpoGA
wwwrun ALL=(ALL) NOPASSWD: /usr/sbin/iptables
```

  WICHTIG:
  Um diese Änderungen vornehmen zu können, sind root-Rechte nötig!!

  Dabei ist die erste Zeile eine Kommentarzeile. `wwwrun` ist der System-User, auf den sich die Änderung auswirken soll.
  `ALL=(ALL)NOPASSWD: /usr/sbin/iptables` bedeutet, dass dieser User grundsätzlich alle Befehle ausführen darf, aber nur den iptables-Befehl auch ohne Passwort-Abfrage. Also z.B. automatisch über ein Skript, wie es in diesem Falle auch benötigt wird. Da man sich auf der Linux-Konsole nicht als „wwwrun“ anmelden kann, wird der Befehl rein dem Apache-Webserver zugänglich gemacht. Weitere Informationen zu diesem Befehl sind im Sudoers-Manual [1] nachzulesen.

- Installation des Firewall-Skriptes
  Das zuvor kurz beschriebene Firewall-Skript muss nun an die richtige Stelle im Datei-System kopiert werden, um auch ausführbar zu sein. Dazu wechselt man in das Verzeichnis, wo sich die SpoGA-Dateien befinden und kopiert die Datei „firewall“ in das Verzeichnis /etc/init.d/ mit folgendem Kommando:

```
spoga:/ # cd srv/www/htdocs/spoga
spoga:/srv/www/htdocs/spoga # cp firewall /etc/init.d/firewall
spoga:/srv/www/htdocs/spoga # cd /
```

[1] Sudoers-Manual: http://www.courtesan.com/sudo/man/sudoers.html

Jetzt muss man es noch ausführbar machen:

```
spoga:/ # chmod +x /etc/init.d/firewall
```

Nun ist das Skript bereit und kann im nächsten Schritt zum Runlevel hinzugefügt werden.

- Anpassen des Runlevels
  Damit die entwickelten Programme bzw. Skripte auch Wirkung zeigen, müssen diese bereits beim Booten des Systems mitgeladen werden. Es gibt mehrere Möglichkeiten, dies zu realisieren. Der wohl komfortabelste Weg ist die Einbindung in die sogenannten Runlevel. Bei jedem System-Start wird in einem bestimmten Runlevel gebootet, in dem dann eine Liste von unterschiedlichen Programmen gestartet wird. Beim Booten des SpoGA-Servers gelangen wir standardmäßig in Runlevel 3. Dabei läuft das System im Mehrbenutzer-Modus mit vollständigen Netzwerk-Funktionen. Der Rechner bootet ohne grafische Oberfläche an einen textbasierten Anmelde-Prompt, wobei alle notwendigen vorkonfigurierten Dienste bereits gestartet wurden. Die meisten Server laufen im Runlevel 3.
  Um nun diese Liste der Programme, die im Runlevel 3 gestartet werden, zu verändern, kommt der Befehl „chkconfig“ zum Einsatz. Mit

```
spoga:/ # /sbin/chkconfig firewall 35
```

wird das Skript in die Runlevel 3 und 5 eingetragen. Runlevel 5 bedeutet, das System bootet mit grafischer Oberfläche. Diese ist zwar nicht installiert, aber es ist sicherer, wenn die Firewall auch in diesem Level läuft, falls später jemand auf die Idee kommt doch eine Grafikoberfläche zu installieren und zu nutzen.

Tauchen nach Aufrufen des Befehls `/sbin/chkconfig --list` folgende Zeilen auf

```
SuSEfirewall2_init 0:off  1:off  2:off  3:off  4:off  5:off  6:
    off  B:on
SuSEfirewall2_setup 0:off  1:off  2:off  3:on   4:on   5:on
    6:off
```

so ist die standardmäßig installierte System-Firewall (in diesem Falle SuSEfirewall2) von SuSE-Linux noch aktiviert, und sollte deaktiviert werden. Dies wird mit den Kommandos

```
spoga:/ # /sbin/chkconfig --level B SuSEfirewall2_init off
spoga:/ # /sbin/chkconfig --level 345 SuSEfirewall2_setup off
```

bewirkt.

Wenn das neue Firewall-Skript irgendwann nicht mehr benutzt werden soll, kann mit

```
spoga:/ # /sbin/chkconfig --del firewall
```

die Änderung wieder rückgängig gemacht werden und die SuSEfirewall kann wieder auf „on“ gesetzt werden. Um den Computer jetzt nicht neu starten zu müssen, erfolgt der erste Start mittels `/etc/init.d/firewall start`

- Nutzen der MAC-Adressen-Filter-Funktion
  Leider ist es im WLAN nicht möglich, die MAC-Adresse eines über mehrere Hops verbundenen Clients abzufragen. Eine solche Funktion (durch Auslesen des arp-Caches) ist im SpoGA-System bei der Anmeldung zwar implementiert, funktioniert aber nicht im wireless LAN.

11 Erweiterungsmöglichkeiten

Das im Rahmen dieser Arbeit realisierte SpoGA-System stellt einen ersten Prototyp dar. Die Architektur des entwickelten Systems ist in Anhang A dargestellt. An einigen Stellen bietet sich unmittelbar die Möglichkeit einer Erweiterung durch neue oder ergänzende Funktionalitäten. Diese sind z.B.:

- Automatisches Abmelden der Nutzer nach einer bestimmten Zeit der Inaktivität (idletime).
  Ist ein Nutzer eine vorgegebene Zeit (bsw. 15 Minuten) nicht aktiv, so könnte er durch ein Skript, das seine Aktivitäten überprüft, automatisch abgemeldet werden. Dies würde den Missbrauch der Gastzugänge durch Dritte einschränken.

- Senden einer Warnmeldung an den Nutzer, wenn dessen Zeitguthaben während einer Sitzung fast abgelaufen ist.
  Kurz vor Ablauf des Zeitguthabens bekommt der Nutzer eine Meldung, dass seine Nutzungszeit abgelaufen ist. Dies verhindert das abrupte Trennen der Verbindung, und ermöglicht dem Anwender, seinen Vorgang im Internet noch zu beenden.

- MAC-Adressen-Filter
  Durch eine Überprüfung der Gastzugänge mittels eine MAC-Adressen-Filter-Funktion wäre eine zusätzliche Authentifizierungsmöglichkeit des Nutzer und eine weitere Hürde für potentielle Angreifer gegeben. Leider war es nicht möglich, eine solche Funktion zu realisieren, da dies nicht von allen gängigen WLAN-Geräten unterstützt wird. Die Funktion ist zwar implementiert, aber somit nicht fürs WLAN einsatzfähig.

- Anbindung eines RADIUS-Servers
  Um bequem und schnell eigene Firmen-Mitarbeiter für die Nutzung des Systems freizuschalten, wäre eine Anbindung eines RADIUS-Servers sinnvoll. Dabei könnten die Clients der Mitarbeit ohne zusätzliche Abfragen die Firewall passieren und Zugang zum SpoGA-System erlangen

In Anbetracht der zeitlichen Begrenzung dieses Projektes wurden diese Erweiterungen bis zum jetzigen Zeitpunkt nicht realisiert.

12 Zusammenfassung

In Zeiten, in denen ein Notebook so selbstverständlich wie ein Taschenrechner ist und als Arbeitsgerät, oder zur Kommunikation bzw. Recherche im Internet genutzt wird, ist es für Gäste von enormem Vorteil, schnell und unkompliziert einen Zugriff auf die vorhandene Netzinfrastruktur einer gastgebenden Firma zu erlangen. Dies erspart einerseitz Arbeitsaufwand von Seiten der Administratoren, bzw. wenn es sich um eine kleinere Firma handelt, die nicht über eine eigene IT-Abteilung verfügt, ermöglicht es, ohne die Dienste von Dritten in Anspruch zu nehmen, einen zeitlich begrenzten Zugang für Gäste. Andererseits lassen sich Kosten für die sonst nötigen Arbeitsschritte einsparen, und die Administratoren können sich ihren eigentlichen Aufgaben widmen. Und das Ganze unabhängig von Arbeits- und Urlaubszeiten, frei von lästigen Formalitäten und ohne Vorlaufzeit, um nur einige der Vorteile gegenüber einer manuellen Freischaltung zu nennen. Ein weiterer wichtiger Punkt ist, dabei die Sicherheit der IT-Netzinfrastruktur nicht zu beeinträchtigen. Ein spontaner Zugang sollte zeitlich begrenzt (z.B. für die Dauer einer Veranstaltung) und personenbezogen sein.

Genau diese Funktionalität ermöglicht das in diesem Projekt entwickelte SpoGA-System. Ein firmeninterner Gastgeber hat dabei die Möglichkeit, ohne einen Administrator kontaktieren zu müssen, Gast-Zugänge zu erstellen und zu verwalten. Vorraussetzung für Gäste ist lediglich ein mobiles Endgerät zum Empfangen eines Zugangscodes per SMS, sowie ein WLAN-fähiges Notebook. Soll eine größere Veranstaltung geplant und dabei Ressourcen wie Räume oder Beamer gebucht werden, so besteht die Möglichkeit, ein optionales System, das E-IMS, einzusetzen. Dieses stellt eine Erweiterung des SpoGA-Systems dar, die in 2 aufbauenden Studienarbeiten realisiert wurde, und ergänzt dessen Funktionalitäten um eine Groupware für Verwaltungs- und Planungsarbeiten.

Abkürzungsverzeichnis

AP	Access Point
E-IMS	Extended Invitation Management System
ISP	Internet Service Provider
MD5	Message Digest Algorithm 5
RZ	Rechenzentrum
SOAP	Simple Object Access Protocol
SpoGA	Spontaneous (WLAN) Guest Access
UDDI	Universal Definition, Discovery and Integration
URL	Uniform Resource Locator
WLAN	Wireless Local Area Network
WSDL	Web Services Description Language
WS	Web Service
XML	eXtensible Markup Language

Abbildungsverzeichnis

Tabellenverzeichnis

Literaturverzeichnis

[And] Andreasson, Oskar: *Iptables Tutorial*. CTAN: http://iptables-tutorial.frozentux.net/chunkyhtml/index.html ,abgerufen am 22.08.2006.

[Ber05] Bergmann, Sebastian: *Professionelle Softwareentwicklung mit PHP 5 - Objektorientierung. Entwurfsmuster. Modellierung. Fortgeschrittene Datenbankprogrammierung.* dpunkt Verlag, 2005.

[CW03] Christian Wenz, Tobias Hauser: *Web Services mit PHP*. Galilio Press, 2003.

[GA04] Gustavo Alonso, Fabio Casati, Harumi Kuno: *Web Services: Concepts, Architecture and Applications*. Springer Verlag, 2004.

[Hor06] Horstmann, Jutta: *Freie Datenbanken im Unternehmenseinsatz*. CTAN: http://www.heise.de/open/artikel/70100/0 ,abgerufen am 04.05.2006, 2006.

[IEE] IEEE: *IEEE 802.11, The Working Group for Wireless LANs*. CTAN: http://grouper.ieee.org/groups/802/11/ ,abgerufen am 22.11.2006.

[IEE01] IEEE: *IEEE Standard for Local and metropolitan area networks, Port-Based Network Access Control*. CTAN: http://standards.ieee.org/getieee802/download/802.1X-2001.pdf ,abgerufen am 17.11.2006, 2001.

[Man06] Manhart, Dr. Klaus: *Web Services - Grundlagen, Aufbau und Struktur*. CTAN: http://www.tecchannel.de/entwicklung/grundlagen/457051/ ,abgerufen am 15.03.2007, 2006.

[Mül07] Müller, Markus: *E-IMS-System*, 2007.

[MyS] MySQL: *The world's most popular open source database*. CTAN: http://www.mysql.com ,abgerufen am 03.05.2006.

[Nov] Novell: *SUSE Linux 10 Reference Guide*. CTAN: http://www.novell.com/documentation/suse10/index.html ,abgerufen am 03.05.2006.

[Nus98] Nusser, Stefan: *Sicherheitskonzepte im WWW*. Springer Verlag, 1998.

[Ope] OpenGroupware.org: *Homepage*. CTAN: http://opengroupware.org ,abgerufen am 06.05.2006.

[phpa] phpMyAdmin: *Documentation*. CTAN: http://www.phpmyadmin.net/documentation/ ,abgerufen am 03.05.2006.

[PHPb] PHP.NET: *Documentation*. CTAN: http://www.php.net/ ,abgerufen am 12.07.2006.

[PS04] Peter Stahlknecht, Ulrich Hasenkamp: *Einführung in die Wirtschaftsinformatik - 11. Auflage*. Springer Verlag, 2004.

[Rec06] Rech, Jörg: *Wireless LANs - 802.11-WLAN-Technologie und praktische Umsetzung im Detail - 2. aktualisierte Auflage.* Heise Verlag, 2006.

[Sch02] Schwenk, Jörg: *Sicherheit und Kryptographie im Internet - Von sicherer E-Mail bis zu IP-Verschlüsselung.* Vieweg Verlag, 2002.

[SEL] SELFHTML: *Documentation - Version 8.1.1.* CTAN: http://www.selfhtml.org/ ,abgerufen am 12.07.2006.

[SMS] SMS77.DE: *Dokumentation der API.* CTAN: http://www.sms77.de/api.pdf ,abgerufen am 22.08.2006.

[Spe07] Speich, Christoph: *E-IMS-System (extended)*, 2007.

[Ste02] Steven, Marion: *BWL für Ingenieure.* Oldenbourg Verlag, 2002.

[Tan03] Tanenbaum, Andrew S.: *Computernetzwerke - 4. überarbeitete Auflage.* Pearson Studium, 2003.

[Wik] Wikipedia: *ISO/IEC 9126 Norm.* CTAN: http://de.wikipedia.org/wiki/ISO_9126 ,abgerufen am 12.01.2007.

A Architektur des erstellten Prototypen

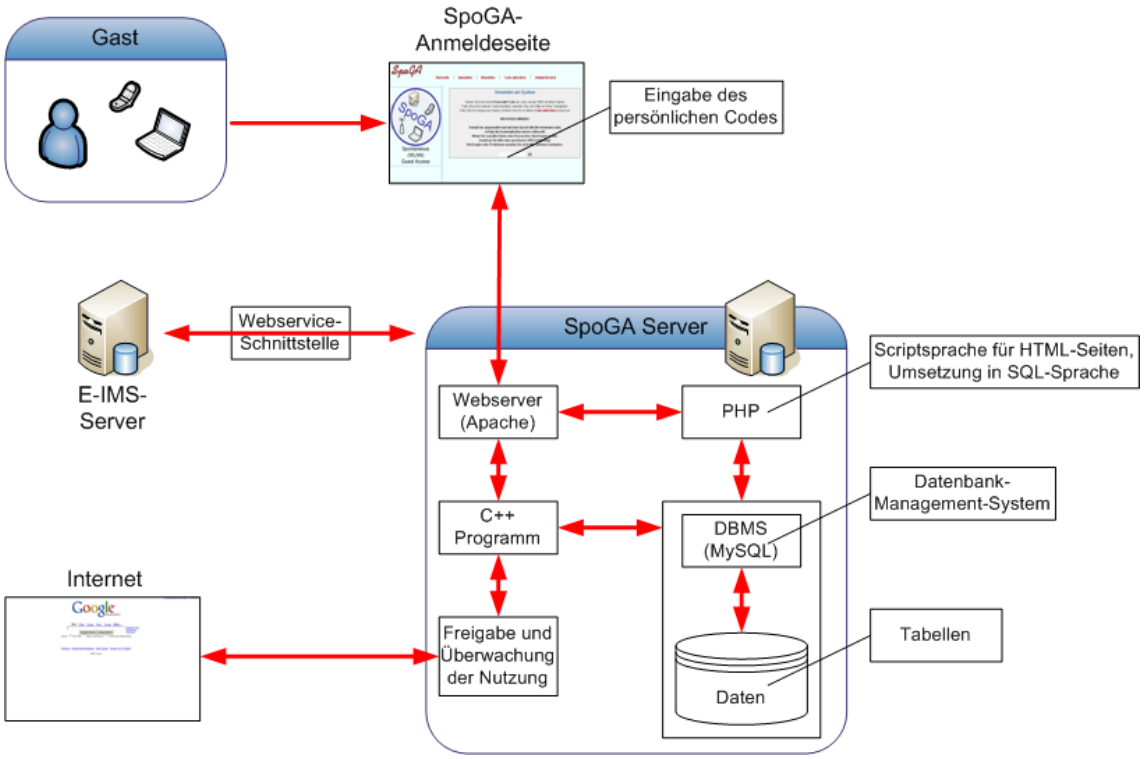


Abbildung A.1: Gesamtarchitektur

B Interview zum Einsatz von Groupware an der Universität Koblenz

B.1 Einleitung

Über die Seiten des Rechenzentrums der Universität Koblenz-Landau wurden wir auf einen aktuell stattfindenden Test von Groupware-Systemen an der Universität Koblenz-Landau, Campus Koblenz aufmerksam. Um von den im Rahmen dieses Testes gesammelten Erfahrungen profitieren zu können, vereinbarten wir mit dem wissenschaftlichen Leiter des Rechenzentrums, Uwe Arndt einen Gesprächstermin. Im folgenden skizzieren wir eine Kurzzusammenfassung dieses Gespräches, welches am 13.03.2006 um 10 Uhr stattfand.

B.2 Das Interview

Frage: Gründe für die Auswahl der beiden Systeme Open-XChange und OpenGroupware?

Herr Arndt:
Das Hauptargument für den Einsatz der beiden oben genannten Systeme ist ihre Verfügbarkeit als OpenSource-Produkte. Open-XChange wurde bereits früher in der zu diesem Zeitpunkt kommerziellen Version SuSE Mail 4 getestet. Für einen Produktiveinsatz als Groupware-Lösung wäre jedoch der kostenintensive Erwerb von Lizenzen notwendig gewesen, so dass diese Möglichkeit nicht weiter verfolgt wurde.

Frage: Welche Anforderungen stellt die Universität an die Groupware-Systeme?

Herr Arndt:
Da bereits ein Großteil der Funktionen einer typischen Groupware-Lösung von anderen, bereits im Einsatz befindlichen Systemen wie dem BSCW-Server[1] abgedeckt wird, war die einzige wirkliche Hauptanforderung an das einzusetzende System die Bereitstellung eines Gruppenkalenders zur gemeinsamen Terminplanung der Teilnehmer eines Projektes.

Frage: Welche Erfahrungen haben Sie bisher mit den beiden Systemen gemacht?

Herr Arndt:
Zwischen den beiden Produkten existieren bezüglich der Funktionalitäten nur minimale Unterschiede. Aufgrund der nutzerfreundlicheren grafischen Oberfläche gibt es aber eine Präferenz für Open-XChange. Bisher wurde die Software nur mit einer geringen Anzahl von Nutzern getestet, daher steht noch der Test unter realen Bedingungen, also insbesondere einer hohen Anzahl an Zugriffen, aus.

[1] http://bscw.fit.fraunhofer.de/

Frage: Traten beim Probe-Betrieb der Systeme besondere Fehler auf?

Herr Arndt:
Gelegentlich kam es bei OpenXChange zu Problemen bei der Synchronisation mit Outlook. Bei OpenGroupware hingegen sind keine erwähnenswerten Fehlfunktionen aufgetreten.

Frage: Welches der beiden Systeme soll in Zukunft produktiv eingesetzt werden?

Herr Arndt:
Geplant ist die Nutzung von OpenXChange in der Universität Koblenz ab dem Sommersemester 2006. OpenGroupware wird nicht zum Einsatz kommen, da uns die Gestaltung der Oberfläche nicht zusagte.

C Dokumentation der SMS HTTP-API

C.1 Auszug aus der HTTP API von sms77.de

HTTP API Version 0.70 Spezifikationen[1]
Datum: 04. März 2006

Durch das HTTP API ermöglichen wir es Ihnen, SMS über das System von SMS77 direkt aus Ihrer Webseite heraus per http-Request zu versenden. Dadurch lassen sich professionelle SMS-Anwendungen realisieren. SMS77 tritt dabei als White-Label-Anbieter auf. Die Preise sind die gleichen wie beim Webinterface von SMS77.

Aktuelle Preisliste: http://www.sms77.de/?content=prices

Wichtig: Alle Texte, die in der Request-URL vorkommen, müssen urlencoded an das HTTP API übergeben. (insbesondere Parameter text, data bei Klingelton, file bei Logo, file1/file2/file3 bei MMS)

1. Spezifikation SMS-Versand

Der http-Request für SMS ist an die URL http://www.sms77.de/gateway/ zu senden. Ihr Passwort können Sie aus Sicherheitsgründen codiert übertragen (MD5). Ihr codiertes Passwort erfahren Sie in den SMS77-Optionen unter Http Api. Das Api erkennt automatisch, ob Sie ein codiertes Passwort oder nicht übertragen haben. Falls Sie eine sichere Verbindung verwenden möchten, können Sie Ihre Requests auch an https://www.sms77.de/gateway/ senden.

...

Beispiele:

```
http://www.sms77.de/gateway/?u=benutzer&p=passwort&to=00491609876543&text=
meine%20erste%20nachricht&type=quality&from=sms77.de
```

```
http://www.sms77.de/gateway/?u=benutzer&p=passwort&to=0402888888&text=mein
e%20erste%20nachricht&type=festnetz&debug=1
```

Aufgrund des Parameters debug=1 wird keine SMS verschickt sondern lediglich entweder eine Fehlermeldung oder „100“ falls die Parameter OK sind ausgegeben.

...

[1] aus dem Mitgliederbereich von http://www.sms77.de

3. Guthaben abfragen

Über das Script unter der URL http://www.sms77.de/gateway/balance.php können Sie Ihr aktuelles Guthaben bei SMS77 abfragen. Zurückgegeben wird entweder 900 (siehe Rückgabewerte) oder Ihr aktuelles Guthaben (z.B. „12.92“)

...

6. Rückgabewerte

Das API liefert dabei bei erfolgreicher Zustellung den Wert 100 im Mime-Typ text/plain zurück. Ansonsten wird einer der folgenden Fehlercodes zurückgegeben:

Fehlercode Beschreibung 100 SMS wurde erfolgreich verschickt
101 Versand an mindestens einen Empfänger fehlgeschlagen
201 Ländercode für diesen SMS-Typ nicht gültig. Bitte als Basic SMS verschicken.
202 Empfängernummer ungültig
300 Bitte Benutzer/Passwort angeben
301 Variable to nicht gesetzt
304 Variable type nicht gesetzt
305 Variable text nicht gesetzt
306 Absendernummer ungültig (nur bei Standard SMS). Diese muss vom Format 0049... sein und eine gültige Handynummer darstellen.
307 Variable url nicht gesetzt
400 type ungültig. Siehe erlaubte Werte oben.
401 Variable text ist zu lang
402 Reload-Sperre: SMS wurde bereits gesendet
403 Text + URL zu lang (größer als 110 Zeichen)
500 Zu wenig Guthaben vorhanden.
600 Carrier Zustellung misslungen
700 Unbekannter Fehler
801 Logodatei nicht angegeben
802 Logodatei existiert nicht
803 Klingelton nicht angegeben
900 Benutzer/Passwort-Kombination falsch
902 http API für diesen Account deaktiviert
903 Server IP ist falsch

...

D Iptables Befehlsübersicht

Regel	**iptables-Befehl**
Anhängen	-P
Löschen	-D
Ersetzen	-R
Einfügen	-I
Regelliste	
Anzeigen	-L
Leeren	-F
Neu	-N
Löschen	-X
Umbenennen	-E
Filtertabelle	
Input	INPUT
Output	OUTPUT
Forwarding	FORWARD
Prerouting	PREROUTING
Postrouting	POSTROUTING
Aktionen	
Erlauben	-j ACCEPT
Ablehnen mit Rückmeldung	-reject-with
Ablehnen ohne Rückmeldung	-j DROP
Tabelle verlassen	-j RETURN
Sprung zu benutzerdef. Tabelle	-j <userdefined chain>
Paket in den Userspace leiten	-j QUEUE
Masquerading	-to-ports
SNAT	-to-source
DNAT	-to-destination
LOG	–log-level
MARK	–set-mark
TOS	–set-tos
MIRROR	-j MIRROR(exp)
Umleiten	–to-ports
Parameter	
Absender (source)	-s
Empfänger (destination)	-d
Protokoll	-p

Interface	
Input-Interface	-i
Output-Interface	-o

Tabelle D.1: Iptables Befehlsübersicht

E Verzeichnisstruktur

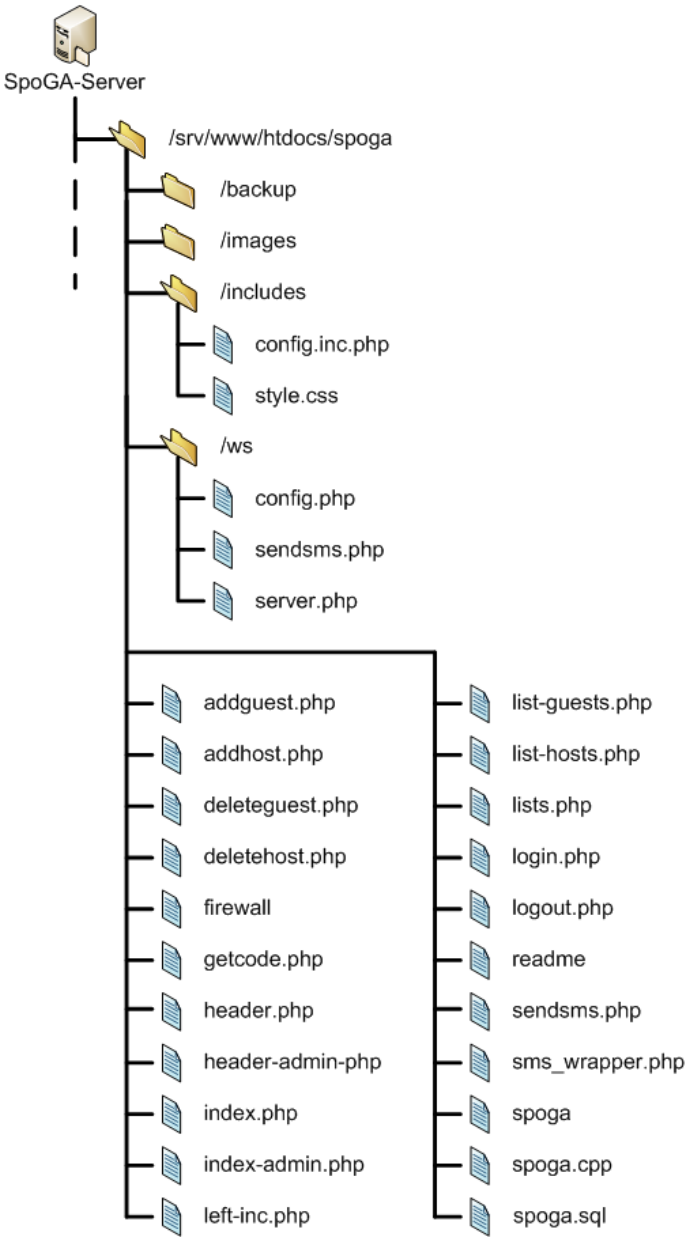


Abbildung E.1: Verzeichnisstruktur der SpoGA-Dateien

F Inhalt der CD

Die beigelegte CD enthält folgende Inhalte:

- **Diese Ausarbeitung als PDF**
- **Diese Ausarbeitung als PS**
- **Diese Ausarbeitung als LaTeX-Source**
- **Quellcode der Programme und Skripte**
- **Quellcode der Web-Oberfläche**
- **MySQL-Importdatei der SpoGA-Datenbanken**
- **Sämtliche Diagramme und Abbildungen dieser Arbeit**

G Source-Code

Nachfolgend eine Auswahl des Source-Codes der wichtigsten entwickelten Programme und Skripte des SpoGA-Servers.

G.1 spoga.cpp

```
/*
 * Programm zur Ueberwachung des SpoGA Servers.
 * Features:
 * Wenn ein User eingeloggt ist werden seine Nutzungszeit und die
     aktuelle Sitzungsdauer ueberwacht
 * Falls die Nutzungszeit abgelaufen ist, oder ein Event
     ausgelaufen ist, wird die IP des Nutzers von der Firewall
     wieder geblockt
  * Eine Stunde vor Eventbeginn bekommt der User eine SMS mit dem
      Freischalt-Code zu seinem aktuellen Event gesendet
 *
 */

#include <stdio.h>
#include <my_global.h>
#include <mysql.h>
#include <stdlib.h>
#include <string>
#include <iostream>
#include <sstream>
#include <time.h>

using namespace std;

int main(){

    cout << "Programm gestartet\n";
    while (true)                //Programm laeuft in Endlosschleife
        , Abbruch durch STRG+C
    {
     sleep(10);                                 //10 Sekunden abwarten
         , dann erst Zugriff auf Datenbank
  cout << time(0);
  cout << ": Abfrage starten...\n";
     MYSQL *verbindung;                         //Variable fÃ¼r MySQL-
         Verbindung
     MYSQL_RES  *mysql_ergebnis1;               //Speichert das
         Ergebnis einer Anfrage
  MYSQL_RES  *mysql_ergebnis2;             //Speichert das Ergebnis
      einer Anfrage
```

```
MYSQL_RES  *mysql_ergebnis3;            //Speichert das Ergebnis
    einer Anfrage
MYSQL_ROW  zeile1;                      //Speichert eine Zeile (=
    Datensatz) des Ergebnisses
MYSQL_ROW  zeile2;                      //Speichert eine Zeile (=
    Datensatz) des Ergebnisses
MYSQL_ROW  zeile3;                      //Speichert eine Zeile (=
    Datensatz) des Ergebnisses
verbindung = mysql_init(NULL);          //Datenbank
    initialisieren
   if(verbindung == NULL)                  //Fehler bei der
       Initialisierung
   {
     fprintf(stderr, "Initialisierung fehlgeschlagen\n");
     exit (0);
   }
   //Verbindung zur Datenbank herstellen (mit Fehlerbehandlung)
   if (mysql_real_connect(verbindung,"spoga.uni-koblenz.de","root
       ","spoga12","spoga",0,NULL,0) == NULL)
   {
    fprintf (stderr, "Fehler mysql_real_connect(): %u (%s)\n",
        mysql_errno (verbindung), mysql_error (verbindung));
   }


/*
 * Behandlung der User, die eingeloggt sind und deren Event
     auseglaufen ist oder deren Nutzungszeit vorueber ist
 * iptables droppen, login = 0 setzen, log-table fuellen
 */

   //Anfrage ausfuehren
   mysql_query(verbindung,"SELECT mobilenr,ip,begin,end,credits,
       duration,whopays FROM jobs WHERE (((UNIX_TIMESTAMP()>= end
        AND end!=0) OR credits<=0) AND login=1)");
//User suchen, die eingeloggt sind und deren Nutzungszeit oder
    Event abgelaufen ist
mysql_ergebnis1 = mysql_store_result(verbindung);          //
    Ergebnis der Anfrage speichern

// iptables Befehle loeschen und login auf 0 setzen
while ((zeile1 = mysql_fetch_row(mysql_ergebnis1)) != NULL)
    //Ergebnis zeilenweise auswerten
   {
 //iptables Befehl 1 loeschen
    string befehl="sudo iptables -D FORWARD -s ";
    befehl.append((string)zeile1[1]);           //IP-Adresse (
        steht in zeile1[1]) in den String "befehl" einfuegen
    befehl.append(" -i eth1 -j ACCEPT");
    system(befehl.c_str());                    //String "befehl"
         nach char* konvertieren, iptables ausfuehren
    //iptables Befehl 2 loeschen
    string befehl2="sudo iptables -D FORWARD -d ";
```

```
    befehl2.append((string)zeile1[1]);
    befehl2.append(" -i eth0 -j ACCEPT");
    system(befehl2.c_str());
    //iptables Befehl 3 loeschen
    string befehl3="sudo iptables -D PREROUTING -t nat -p tcp -s
        ";
    befehl3.append((string)zeile1[1]);
    befehl3.append(" --dport 80 -j ACCEPT");
    system(befehl3.c_str());

    //Anfrage zum Updaten des aktuellen Datensatz erstellen:
        Anfangs- und Endezeit sowie Dauer auf "0" setzen,
 // da Zeit abgelaufen

    string anfrage1="UPDATE jobs SET login=0 WHERE (mobilenr=";
    anfrage1.append((string)zeile1[0]);
 anfrage1.append(" AND login=1)");
    mysql_query(verbindung,anfrage1.c_str());            //
        Anfrage aus String "anfrage" in char* konvertieren und an
         DB senden

 // Log-Datei fuellen

    string anfrage2="UPDATE log SET log_end=UNIX_TIMESTAMP(),log_
        duration=";
 anfrage2.append((string)zeile1[5]); // duration
 anfrage2.append(" WHERE (log_mobilenr=");
 anfrage2.append((string)zeile1[0]); // mobilenr
 anfrage2.append(" AND log_end=0)");
 mysql_query(verbindung,anfrage2.c_str());           //Anfrage
     aus String "anfrage" in char* konvertieren und an DB senden

 // DEBUGGING
 //cout << "1-Anfrage1"+anfrage1+"\n";
 //cout << "1-Anfrage2"+anfrage2+"\n";
 //
    //if (mysql_affected_rows(verbindung)!=1)
 //{           // Datensatz unveraendert
    // cout << "Datensatz wurde nicht veraendert!\n";
 //}

}


/*
 * Behandlung der User, die eingeloggt sind
 * surftime erhoehen, duration erhoehen, credits abziehen
 */


mysql_query(verbindung,"SELECT mobilenr,ip,begin,end,duration,
    surftime,credits FROM jobs WHERE (login=1)");   // User
    suchen, die eingeloggt sind
```

```
mysql_ergebnis2 = mysql_store_result(verbindung);       //
    Ergebnis der Anfrage speichern

while ((zeile2 = mysql_fetch_row(mysql_ergebnis2)) != NULL)
           //Ergebnis zeilenweise auswerten
   {

 string anfrage="UPDATE jobs SET duration=duration+10, surftime=
     surftime+10,credits=credits-10 WHERE (mobilenr=";
 anfrage.append((string)zeile2[0]);
 anfrage.append(" AND login=1)");
 mysql_query(verbindung,anfrage.c_str());            //Anfrage
     aus String "anfrage" in char* konvertieren und an DB senden

 // DEBUGGING
 //cout << "2-Anfrage1"+anfrage+"\n";
 //
 //if (mysql_affected_rows(verbindung)!=1)
 //{           // Datensatz unveraendert
 // cout << "Datensatz wurde nicht veraendert!\n";
 //}
}

/*
 * Behandlung der User, die 1h vor eventbeginn eine SMS bekommen
     sollen mit dem Code
 */

mysql_query(verbindung,"SELECT mobilenr,code,begin FROM jobs
    WHERE ((begin-3600)<(UNIX_TIMESTAMP())) AND ((begin-3589)>(
    UNIX_TIMESTAMP()))");
mysql_ergebnis3 = mysql_store_result(verbindung);       //
    Ergebnis der Anfrage speichern

while ((zeile3 = mysql_fetch_row(mysql_ergebnis3)) != NULL)
           //Ergebnis zeilenweise auswerten
   {
 // SMS Senden
 // mittels sms_wropper.php
 // values: nr,text
 // nr=mobilenr aus table jobs, ohne fuehrendes + und mit 00
     stattdessen
 // format der nr muss sein: 0049170......

 string nr=(string)zeile3[0];
 nr.replace(0, 1, "00");      // mobilenr konvertieren in format
     0049170.....
 string befehl="php -f sms_wrapper.php ";
    befehl.append(nr);
    befehl.append(" 'Ihr Event startet ihn 1h. Der Code lautet:
        ");
    befehl.append((string)zeile3[1]);          // hier steht der
        code
```

```
   befehl.append(" und wird in 1h freigeschaltet. Ihr SpoGA-Team'")
       ;
   cout << befehl+"\n";
   system(befehl.c_str());                    //String "befehl"
       nach char* konvertieren und als systembefehle ausfuehren

     }

  mysql_free_result(mysql_ergebnis1);          //Speicher fÃ¼r
      Ergebnisse freigeben
     mysql_free_result(mysql_ergebnis2);          //Speicher fÃ¼r
         Ergebnisse freigeben
     mysql_free_result(mysql_ergebnis3);          //Speicher fÃ¼r
         Ergebnisse freigeben
  mysql_close(verbindung);                     //Verbindung beenden
     cout << "Abfrage beendet...\n";
    }
    cout << "Programm beendet\n";
    return 0;
}
```

G.2 Firewall

```
##!/bin/sh
#
#  Dieses script in /etc/init.d kopieren
#

set -e

PATH=/usr/local/sbin:/usr/local/bin:/sbin:/bin:/usr/sbin:/usr/bin
DESC="Firewall"
NAME=SpoGA_Firewall
DAEMON=/usr/sbin/$NAME
PIDFILE=/var/run/$NAME.pid
SCRIPTNAME=/etc/init.d/$NAME

d_start() {
 start-stop-daemon --start --quiet --pidfile $PIDFILE \
  --exec $DAEMON

}

d_stop() {
 start-stop-daemon --stop --quiet --pidfile $PIDFILE \
  --name $NAME
 iptables -F                    #alle Regeln loeschen
 iptables -A INPUT -j ACCEPT    #alle ankommenden Pakete erlauben
 iptables -A OUTPUT -j ACCEPT   #alle ausgehenden Pakete erlauben
 iptables -A FORWARD -j ACCEPT  #alle weiterzuleitenden Pakete
     erlauben
}

d_reload() {
 start-stop-daemon --stop --quiet --pidfile $PIDFILE \
  --name $NAME --signal 1
}

case "$1" in
  start)
 #
 # hier die iptables-Regel festlegen
 # richtige IP-Adresse beachten!
 #
 echo -n "Starting $DESC: $NAME"
 iptables -F
 iptables -A INPUT -s 0/0 -d 0/0 -i lo -j ACCEPT
  iptables -A OUTPUT -s 0/0 -d 0/0 -o lo -j ACCEPT

 #Port 53 (DNS) erlauben und weiterleiten
 iptables -A OUTPUT -o eth0 -p tcp --destination-port 53 -j ACCEPT
 iptables -A OUTPUT -o eth0 -p udp --destination-port 53 -j ACCEPT
 iptables -A OUTPUT -o eth1 -p tcp -s 192.168.10.1 --sport 53 -j
     ACCEPT
```

```
iptables -A OUTPUT -o eth1 -p udp -s 192.168.10.1 --sport 53 -j
    ACCEPT
iptables -A INPUT -i eth0 -p tcp --source-port 53 -m state --state
     ESTABLISHED,RELATED -j ACCEPT
iptables -A INPUT -i eth0 -p udp --source-port 53 -m state --state
     ESTABLISHED,RELATED -j ACCEPT
iptables -A INPUT -i eth1 -p tcp --destination-port 53 -j ACCEPT
 iptables -A INPUT -i eth1 -p udp --destination-port 53 -j ACCEPT
iptables -A FORWARD -i eth1 -p tcp --destination-port 53 -j ACCEPT
iptables -A FORWARD -i eth0 -p tcp --source-port 53 -m state --
    state ESTABLISHED,RELATED -j ACCEPT

#Port 80 erlauben und weiterleiten
iptables -A OUTPUT -o eth0 -p tcp --destination-port 80 -j ACCEPT
   iptables -A OUTPUT -o eth1 -p tcp --source-port 80 -m state --
       state ESTABLISHED,RELATED -j ACCEPT
iptables -A INPUT -i eth1 -p tcp --destination-port 80 -j ACCEPT
iptables -A INPUT -i eth0 -p tcp --source-port 80 -m state --state
     ESTABLISHED,RELATED -j ACCEPT
iptables -A FORWARD -i eth1 -p tcp --destination-port 80 -j ACCEPT
iptables -A FORWARD -i eth0 -p tcp --source-port 80 -m state --
    state ESTABLISHED,RELATED -j ACCEPT

#VPN Ports weiterleiten
iptables -A OUTPUT -o eth0 -p tcp --destination-port 1723 -j
    ACCEPT
   iptables -A OUTPUT -o eth1 -p tcp --source-port 1723 -m state
       --state ESTABLISHED,RELATED -j ACCEPT
iptables -A INPUT -i eth1 -p tcp --destination-port 1723 -j ACCEPT
iptables -A INPUT -i eth0 -p tcp --source-port 1723 -m state --
    state ESTABLISHED,RELATED -j ACCEPT
iptables -A FORWARD -i eth1 -p tcp --destination-port 1723 -j
    ACCEPT
iptables -A FORWARD -i eth0 -p tcp --source-port 1723 -m state --
    state ESTABLISHED,RELATED -j ACCEPT
iptables -A OUTPUT -o eth0 -p udp --destination-port 47 -j ACCEPT
   iptables -A OUTPUT -o eth1 -p udp --source-port 47 -m state --
       state ESTABLISHED,RELATED -j ACCEPT
iptables -A INPUT -i eth1 -p udp --destination-port 47 -j ACCEPT
iptables -A INPUT -i eth0 -p udp --source-port 47 -m state --state
     ESTABLISHED,RELATED -j ACCEPT
iptables -A FORWARD -i eth1 -p udp --destination-port 47 -j ACCEPT
iptables -A FORWARD -i eth0 -p udp --source-port 47 -m state --
    state ESTABLISHED,RELATED -j ACCEPT

#Umleitung von nicht angemeldeten Usern auf Startseite
iptables -A PREROUTING -t nat -p tcp --dport 80 -j DNAT --to-
    destination 192.168.10.1:80
iptables -A INPUT -j DROP
iptables -A OUTPUT -j DROP
iptables -A FORWARD -j DROP
#d_start
echo "."
```

```
 ;;
  stop)
 echo -n "Stopping $DESC: $NAME"
 iptables -F
 #d_stop
 echo "."
 ;;
 cloak)
 echo -n "Deny all"
 iptables -A INPUT -j DROP
   iptables -A OUTPUT -j DROP
 iptables -A FORWARD -j DROP
 ;;
 restart)
 echo -n "Restarting $DESC: $NAME"
 d_stop
 sleep 1
 d_start
 echo "."
 ;;
  status)
 pids=‘ps ax|grep "firewall"|grep -v grep|cut -d" " -f2‘
 if test "$pids"
 then
  for p in $pids
  do
   echo "Das Programm laeuft als Prozess $pids"
  done
 else
  echo "Das Programm ist gestoppt."
 fi
 ;;
  *)
 echo "Usage: $SCRIPTNAME {start|stop|cloak|restart|status}" >&2
 exit 1
 ;;
esac
exit 0
```

G.3 Webservice-Server

```
<?PHP

$ipadresse = $_SERVER['REMOTE_ADDR'];

require_once("nusoap.php");

$s=new soap_server;

$s->register('senddatabe');  // Events von Groupware in Job-Tabelle
     eintragen
$s->register('com_test');  //Kommunikationstest
$s->register('send_sms');  // SMS-Sendefunktion
$s->register('send_bill');      // Daten fuers billing
$s->register('update_number'); // zum updaten der Handynummer


/*
 * Namenskonventionen:
 * - duration: Surfdauer der aktuellen Sitzung, Differenz zwischen
     Begin und End in der DB "spoga"
 * - credits: Zeit(Guthaben) die ein User hat, um zwischen begin/
     end eines Events zu surfen
 * - surftime: gesamte "versurfte" Zeit, also die Summe aller
     bisherigen durations
 *   wird für das Billing benötigt!
 */

function senddatabe($handynr,$begin,$end,$managerid,$server_passwd
    ,$checksum,$duration,$whopays)
{            //whopays = Kostenträger(int), 0=free 1=everbody
    himself 2= eventcreator 9=from spoga page
 include_once("config.php");
 if ($server_passwd == md5($serverpasswd))
 {
  $sum = $handynr[1]+$handynr[2]+$handynr[4]+$handynr[5]+$handynr
      [6]+$handynr[8]+$handynr[9];
  $tmp = ($begin+$end+$duration+$sum+$whopays);
  $testsum = md5($tmp);
  if ($testsum == $checksum)
  {
   $link = mysql_connect($mysql_host, $mysql_user, $mysql_passwd);
   if (!$link)
   {
    return "Fehler beim Verbindungsaufbau";
   }
   else
   {
    mysql_select_db($mysql_db) or die("Auswahl der Datenbank
        fehlgeschlagen!");
    $dateArray = getdate();
```

```
    $tmp = $dateArray[0] + 4711 ;   // geheimer Schlüssel für den
        Code = Aktueller Unix-Timestamp + 4711
    $codetmp = md5($tmp);     // verschlüsselung des Code
    $code = substr($codetmp, 0, 8);   // Nur die ersten 8 Stellen
        des Code
    $credits = $duration;
    $query = "INSERT INTO jobs (mobilenr,begin,end,manager_id,
        whopays,surftime,duration,ip,login,credits,code)
               VALUES('".$handynr."','".$begin."','".$end."','".$
                   managerid."','".$whopays."',0,0,'0.0.0.0',0,'".$
                   credits."','".$code."')";
    $result=mysql_query($query);
    //$result = true;
    if (!$result)
    {
     return "No Data";
    }
    else
    {
     return "OK";
    }
    mysql_close($link);
   }
  }
  else return "Wrong Checksum";
 }
 else return "Wrong Serverpassword";
}

function com_test($server_passwd)
{
 include_once("config.php");
 if ($server_passwd == md5($serverpasswd))
 {
  return "COM_OK";
 }
 else return "Wrong Serverpassword";
}

function send_sms ($manager_id)
{
 // SMS versenden
 // Bitte die Config-Datei editieren um es "scharf" zu machen!
 include_once("config.php");
 include_once("sendsms.php");       // Datei zum Senden der SMS
     einbinden
 $link = mysql_connect($mysql_host,$mysql_user,$mysql_passwd);
 if (!$link)
 {
  return "Fehler beim Verbindungsaufbau";
 }
 else
 {
```

```
mysql_select_db($mysql_db) or die("Auswahl der Datenbank
    fehlgeschlagen!");
$query = "SELECT * FROM jobs WHERE manager_id = ".$manager_id;
$result=mysql_query($query);
$daten=mysql_fetch_array($result, MYSQL_ASSOC);
if (!$result)
{
 return "No data";
}
else
{
 $nr = $daten['mobilenr'];
 $sms_receiver = str_replace("+","00",$nr);  // + aus der Handynr
      entfernen für richtiges Versenden per Gateway
 if ($daten['whopays']==0) // Kostenträger = free surfing
 {
  $sms_text = "Ihr Freischaltcode für das SpoGA System lautet :
      ".$daten['code']." Sie surfen for free! Viel Vergnügen mit
      unserem Dienst!";
 }
 if ($daten['whopays']==1) // Kostenträge = everbody himself
 {
  $sms_text = "Ihr Freischaltcode für das SpoGA System lautet :
      ".$daten['code']." Sie surfen auf Ihre eigenen Kosten! Viel
       Vergnügen mit unserem Dienst!";
 }
 if ($daten['whopays']==2) // Kostenträge = event creator
 {
  $sms_text = "Ihr Freischaltcode für das SpoGA System lautet :
      ".$daten['code']." Sie surfen auf Kosten des Events! Viel
      Vergnügen mit unserem Dienst!";
 }
 if ($daten['whopays']==9) // Kostenträge = SpoGA page
 {
  $sms_text = "Ihr Freischaltcode für das SpoGA System lautet :
      ".$daten['code']." Sie surfen auf Kosten des Gastgebers!
      Viel Vergnügen mit unserem Dienst!";
 }
 $length=strlen($sms_text);
 if ($length <= $sms_maxlength)    // Abfrage, ob der SMS-Text
     nicht zu lang ist
 {
  $sms_to = $sms_receiver;
  $sms_msg = $sms_text;
  $msg = urlencode($sms_msg);    // Textnachricht wird für URL-
      Übergabe codiert
  $url = 'http://sms77.de/gateway/?u='.$sms_user.'&p='.$sms_
      passwd.'&to='.$sms_to.'&text='.$msg.'&type=basicplus&from
      ='.$sms_from.'';
  $ret = @file($url);      // hier erfolgt der Aufruf des HTTP-
      APIs mittels http-Request
            // das @ ist erforderlich, damit die URL bei Fehler
                nicht mit ausgegeben wird
```

```
    if ($ret[0] == "100")
    {
     return "OK";
    }
    else
    {
     return "Fehler beim SMS-Versand! Fehlercode: ".$ret[0]; //
         Fehlercodeausgabe
    }
    /*$do = send($sms_receiver,$sms_text);
    return "OK";
    if (!$do)
    {
     return "Fehler beim Sender der SMS";
    }
    else return "OKI";*/
   }
   else return "Text der SMS zu lang";
  }
 }
 mysql_close($link);
}

function send_bill ($manager_id, $checksum, $server_passwd)
{
 include_once("config.php");
 $tmp = $manager_id + $manager_id;
 if (md5($tmp) == $checksum)
 {
  if ($server_passwd == md5($serverpasswd))
  {
   $link = mysql_connect($mysql_host,$mysql_user,$mysql_passwd);
   if (!$link)
   {
    return "Fehler beim Verbindungsaufbau";
   }
   else
   {
    mysql_select_db($mysql_db) or die("Auswahl der Datenbank
        fehlgeschlagen!");
    $query = "SELECT surftime FROM jobs WHERE manager_id = ".$
        manager_id;
    $result=mysql_query($query);
    $daten=mysql_fetch_array($result, MYSQL_ASSOC);
    if (!$result)
    {
     return "No data";
    }
    else return $daten['surftime'];
    mysql_close($link);
   }
  }
  else return "Wrong Serverpassword";
```

```
 }
 else return "Wrong Checksum";
}

function update_number($manager_id, $mobile_number, $checksum, $
    server_passwd)
{
 include_once("config.php");
 $tmp = $manager_id + $mobile_number;
 if (md5($tmp) == $checksum)
 {
  if ($server_passwd == md5($serverpasswd))
  {
   $link = mysql_connect($mysql_host,$mysql_user,$mysql_passwd);
   if (!$link)
   {
    return "Fehler beim Verbindungsaufbau";
   }
   else
   {
    mysql_select_db($mysql_db) or die("Auswahl der Datenbank
        fehlgeschlagen!");
    $query = "UPDATE jobs SET mobilenr = '".$mobile_number."' WHERE
         manager_id = ".$manager_id;
    $result=mysql_query($query);
    if (!$result)
    {
     return "No data";
    }
    else return "OK";
    mysql_close($link);
   }
  }
  else return "Wrong Serverpassword";
 }
 else return "Wrong Checksum";
}
$s->service($HTTP_RAW_POST_DATA);
?>
```

G.4 sendsms.php

```
<?
/*
  * Script zum Senden von SMS ueber SMS-Gateway (hier SMS77.de)
  * config-Datei mit weiteren Variablen wird benoetigt
  */

function send($sms_to,$sms_msg)
{
 include("includes/config.inc.php");  // Konfigurationsdatei
     einbinden
 $msg = urlencode($sms_msg);    // Textnachricht wird für URL-
     Übergabe codiert
 $url = 'http://sms77.de/gateway/?u='.$sms_user.'&p='.$sms_passwd
     .'&to='.$sms_to.'&text='.$msg.'&type=basicplus&from='.$sms_
     from.'';
 $ret = @file($url);      // hier erfolgt der Aufruf des HTTP-APIs
     mittels http-Request
           // das @ ist erforderlich, damit die URL bei Fehler
               nicht mit ausgegeben wird
 if ($ret[0] == "100")
 {
  echo "SMS erfolgreich versendet!\n";
  return true;
 }
 else
 {
  echo "Fehler beim SMS-Versand! Fehlercode: ".$ret[0]."\n"; //
      Fehlercodeausgabe
  return false;
 }
}
?>
```

G.5 sms-wrapper.php

```
<?php
/*
  * Wrapper-Script um send() aus dem C++ Programm heraus
      auzufuehren
  * command: php -f sms_wrapper 00491751111111 'Text der SMS...'
  * Format der Handynr ist 0049170.... oder 49170.... oder 0170...
  */

require_once 'sendsms.php';
echo send($argv[1],$argv[2]);
?>
```