Städtisches Siebengebirgsgymnasium 2008/2009 Bad Honnef

Entwicklung und Evaluation eines Systems zur Durchführung von Umfragen in lokalen Netzwerken und Intranets auf Basis des Client-Server-Modells

Facharbeit

eingereicht bei
Herrn Kaibel

vorgelegt von
Maximilian Strauch

Kurs Informatik 1

erstellt
vom 13.02. bis zum 27.03.2009

Inhaltsverzeichnis

1. Abstract

Die vorliegende wissenschaftliche Arbeit beinhaltet die Entwicklung, Bearbeitung und Evaluation eines Systems zur Durchführung von Umfragen in lokalen Netzwerken bzw. Intranets.
Die Idee entstand aus dem Bedarf, eine unbeschränkte Anzahl von Umfragen mit einer unbeschränkten Anzahl von Teilnehmern durchführen und auswerten zu lassen. Die Systemarchitektur basiert auf dem Client-Server- Modell. Entwickelt wurde das System mit der plattformunabhängigen Programmiersprache Java.
Die fertige Anwendungssoftware konnte im Rahmen der Arbeit mit Unterstützung von 26+1 Testpersonen anhand eines Betatests auf Funktionalität und Design geprüft werden.
Dabei wurden insgesamt die Funktionalität und das Design - mit kleineren Abstrichen und begrenztem Nachbesserungsbedarf – bestätigt. Die Evaluation ergab zudem Erkenntnisse über mögliche Verwertbarkeit und Fortentwicklung der Programminhalte.
Zusammenfassend wird mit der Arbeit eine erfolgreiche Softwarearchitektur von der Idee bis zur Anwendungsreife vorgestellt.

2. Einleitung

2.1 Ziel der Facharbeit

Das Ziel dieser Facharbeit ist es, die Entwicklung und den Aufbau eines plattformunabhängigen Umfragesystems zu zeigen und dieses mittels eines Praxistests zu evaluieren.

2.2 Einführung in das Thema

2.2.1 Netzwerke

Definition

Als Netzwerke werden Datenkommunikationssysteme bezeichnet, die durch Übertragung physikalischer Signale den Datenaustausch zwischen mehreren unabhängigen Geräten ermöglichen. Im Allgemeinen unterscheidet man die offenen Netzwerke, die den weltweiten Zugriff ermöglichen, von den geschlossenen Netzwerken, die nur einem bestimmten Nutzerkreis vorbehalten sind. Die Klassifizierung von Netzwerken kann nach der Art Ihrer Verknüpfungen, nach der Zugriffsart oder der räumlichen Ausdehnung geschehen (F.A. Brockhaus GmbH, 1999).

Kurzer Einblick in die Geschichte der Netzwerke

In den 60er Jahren entwickelten sich die ersten großen Netzwerke bzw. Vorläufer des Internets. Die weltpolitische Lage dieser Zeit war für die Entwicklung von großer Wichtigkeit; es war die Zeit des „Kalten Krieges“ zwischen den Weltmächten USA und UdSSR.

Im „Department of Defense“[1] suchte man eine Lösung für ein drohendes Problem: die militärischen Daten sollten bei einem atomaren Angriff des Gegners nicht zerstört werden. Die betreffenden Daten sollten an verschiedensten Standorten gesichert werden, die über das Land verteilt sind und mit einander vernetzt sind. Die Rechner der einzelnen Standorte sollten sich bei Änderungen binnen kürzester Zeit aktualisieren. Dadurch, dass die Rechner auf verschiedenen Wegen mit jedem anderen verbunden waren, sollte im Falle einer Zerstörung eines Standorts trotzdem noch eine Kommunikation – und somit der Zugriff auf die Daten – möglich sein.

Nach einigen Jahren scheiterte das Projekt, wurde aber 1966 von der ARPA[2] zur Vernetzung der ARPA-eigenen Großrechner wiederbelebt. 1969 waren die ersten vier ARPA-Rechner ver-

[1] Amerikanisches Verteidigungsministerium

[2] „Advanced Research Projects Agency“: seit 1958 bestehende, wissenschaftliche Einrichtung zur militärischen Forschung

netzt; knapp drei Jahre später waren bereits 40 Rechner vernetzt. Aus dem ARPA-Netz entwickelten sich später das Internet und andere nicht militärische Netzwerke (Münz, et al., 2005).

Die Protokolle – ein wichtiger Teil des Netzwerks

In Netzwerken nehmen Protokolle, gleichgültig welcher Art, einen wichtigen Stellenwert ein. Sie regeln die Verbindung und den Datenaustausch zwischen Rechnern. Dort ist festgelegt, wie die Kommunikation abläuft oder beispielsweise beendet wird. Auch die Kommunikation zwischen Menschen findet nach einem Protokoll statt: die Grammatik, welche u.a. die Satzstruktur beinhaltet. Auch diese wissenschaftliche Arbeit ist nach einem bestimmten Protokoll abgefasst.

Das TCP/IP-Protokoll[3] ist das wichtigste Protokoll für die Kommunikation im Internet. Das HTTP-Protokoll[4] ist vermutlich das bekannteste Protokoll und regelt die Übertragung von Dateien im Internet. Ohne diese beiden Protokolle wäre es nicht möglich, im Internet zu „surfen“ und es könnte also keine Kommunikation zwischen Rechnern und Menschen statt finden. Die Netzwerktechnolgie könnte nicht genutzt werden (Münz, et al., 2005).

Ein weiterer wichtiger Aspekt bei Protokollen ist die Sicherheit. Zum Einen die Sicherheit, dass keiner die Übertagung „mitlesen“ kann und zum Anderen, dass die Übertragung sicher – also fehlerfrei – vom Sender zum Empfänger übertragen wird.

Das TCP/IP-Protokoll gewährleistet diese Sicherheiten – vor allem aber letztere. Die Daten werden mit dem TCP-Protokoll in Pakete verpackt und gesendet. Das IP-Protokoll stellt sicher, dass die Sendung zum Empfänger übertragen wird. Am Ziel angekommen, werden die Pakete gesammelt und mit dem TCP-Protokoll wieder zusammen gesetzt. Falls einige Pakete verloren gehen, werden diese nochmals übertragen.

2.2.2 Vote

„Vote“ ist das plattformunabhängige System zur Durchführung von Umfragen in lokalen Netzwerken, das aus dieser wissenschaftlichen Arbeit hervor geht. Vote ist speziell konzipiert für Schulnetzwerke, da es im Rahmen einer schulischen Arbeit entwickelt wurde. Es kann aber auch in anderen lokalen Netzwerken verwendet werden.

Vote beschränkt sich auf lokale Netzwerke (LAN), da die Einbindung größerer Netzwerke oder gar des Internets sehr kompliziert ist. So ist es zum Beispiel schwierig, über das Internet eine Kommunikation aufzubauen, denn diverse Firewalls und Router blockieren diese Kommunikation, da es sich nicht um eine Standardverbindung (auf einem Standardport im lokalen Netz-

[3] TCP: „Transmission Control Protocol“/IP: „Internet Protocol“
[4] HTTP: „Hypertext Transfer Protocol“

werk) handelt. Da Vote dann vom Internet aus erreichbar wäre, müssten zusätzliche Maßnahmen getroffen werden, um Angreifer abzuwehren.

Vote basiert auf der Netzwerkkommunikation via Sockets, die im lokalen Netzwerk ungehindert und sicher ablaufen können. Die ursprünglich geplante Datenbankanbindung wurde weggelassen, da diese in Java relativ umfangreich und schwer zu implementieren ist. Für eine sichere Kommunikation zwischen Datenbank und Programm ist eine komplizierte Architektur verschiedenster Server (z.B. eine 3-Schicht-Architektur u.a. mit Anwendungsserver) nötig, welche den Rahmen dieser wissenschaftlichen Arbeit sprengen würde und außerdem von dem Hauptthema – einem Umfrageserver im lokalen Netzwerk – ablenken würde.

3. Entwicklung

Das System Vote teilt sich auf in eine Server-Applikation („VoteServer“), zur Umfragen- sowie Benutzerverwaltung, und eine Client-Applikation („VoteClient“), zur Teilnahme an Umfragen; Vote ist nach dem Client-Server-Modell aufgebaut, doch dazu mehr unter Gliederungspunkt 3.2.1.

3.1 Use-Case

Im Use-Case-Diagramm in Abbildung 1 sind die einzelnen Funktionen des Vote-Systems erkennbar. Das Vote-System ist für das Schulnetzwerk entwickelt worden. So wird der Administrator als Lehrer dargestellt und der Benutzer als Schüler. Administrator des Systems Vote kann jeder sein, der den VoteServer betreibt und somit Umfragen zur Verfügung stellt. Daher muss an dieser Stelle differenziert werden zwischen System- bzw. Netzwerkadministratoren und normalen Benutzern, die im System Vote zu Administratoren werden, da sie zum Beispiel Umfragen bereitstellen.

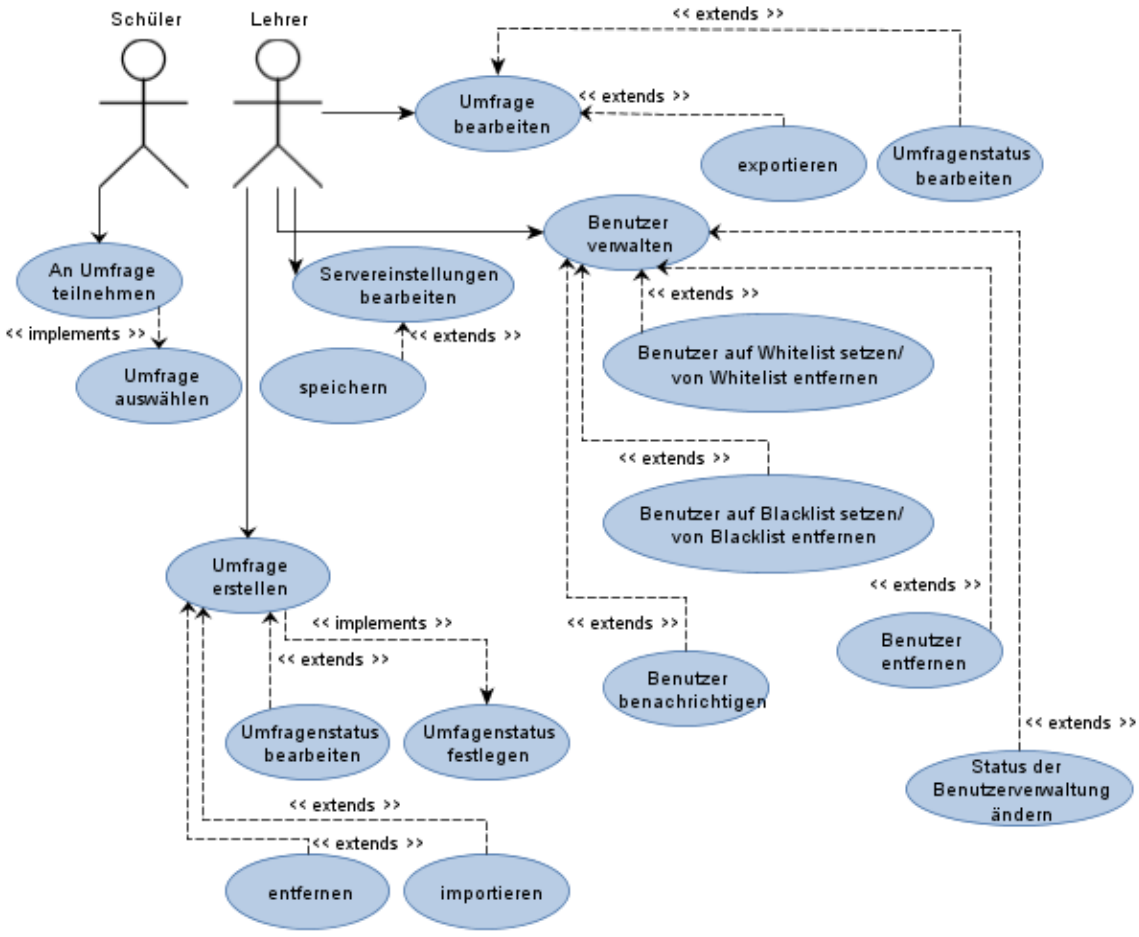


Abbildung 1 Use-Case-Diagramm des Systems Vote

3.2 Softwarearchitektur

3.2.1 Architektur nach dem Client-Server-Modell

Das Client-Server-Modell setzt sich aus zwei Akteuren zusammen: dem Server und dem Client. Server sind Programme, die bestimmte Dienste bereitstellen. Sie warten permanent darauf, dass eine Anfrage eintrifft, die Ihren Dienst betrifft. Der Client nimmt diesen Dienst in Anspruch (Münz, et al., 2005).

Der Aufruf einer Website läuft beispielsweise genau nach diesem Schema ab. Wenn ein Nutzer (Client) eine Website in seinem Browser aufruft, wird der aufzurufende Server kontaktiert. Mittels des HTTP-Protokolls wird in einem sogenannten „HTTP-Request“ (also einer Anfrage) eine Internetseite vom Server abgefragt. Diese richtet sich an den Webserver – ein Dienst, der auf dem Server läuft und dazu dient, Webseiten bereitzustellen. Der Webserver öffnet die Seite, die auf der Festplatte des Servers abgelegt ist und sendet diese in einer „HTTP-Response“ (also einer Antwort) an den Nutzer zurück. Der Browser des Nutzers stellt die erhaltene Webseite dann dar.

Das Client-Server-Modell eignet sich für Vote, da Vote sich in zwei verschiedene Nutzergruppen differenzieren lässt: jene Nutzergruppe, die Umfragen erstellt (also Administratoren des Systems Vote) und durchführt und jene, die an Umfragen teil nimmt.
Genau wie ein Webserver stellt auch der VoteServer Inhalte bereit und hat daneben erweiterte administrative Funktionen, wie zum Beispiel das Aussperren von Nutzern via einer Black- oder Whitelist.
Der VoteClient verhält sich wie ein Client im Client-Server-Modell und kann an Umfragen teil nehmen, also den Dienst des Vote-Servers in Anspruch nehmen.

3.2.2 Schnittstellen der Netzwerkklassen des Zentralabiturs

Java stellt nativ zwei Hauptklassen bereit, um Netzwerkprojekte zu realisieren: ein „ServerSocket“ für einen Server und ein normales „Socket“ für einen Client. Diese Klassen stellen jedoch nur das Werkzeug bereit – die Logik für etwaige Programme muss noch implementiert werden. Da die Entwicklung fehlerfrei funktionierender Klassen für den Server und Client den Rahmen dieser wissenschaftlichen Arbeit gesprengt hätte, wurden für Vote die fertigen Netzwerkschnittstellen des Zentralabiturs verwendet. Diese Netzwerkschnittstellen beinhalten alle nötigen Funktionen, um diverse Applikationen damit zu betreiben.
Im Folgenden werden anhand von Klassendiagrammen die Klassen dargestellt.

Die Serverschnittstelle

Dieser Schnittstelle obliegen zwei wichtige Aufgaben: zum Einen müssen Ereignisse abgefangen werden, wie das An- oder Abmelden eines Clients und das Übertragen einer Nachricht von einem Client. Zum Anderen müssen Nachrichten an Clients gesendet werden können und es muss zusätzlich möglich sein, Clients zu beenden. Auf dem folgenden Klassendiagramm ist die Serverschnittstelle abgebildet:

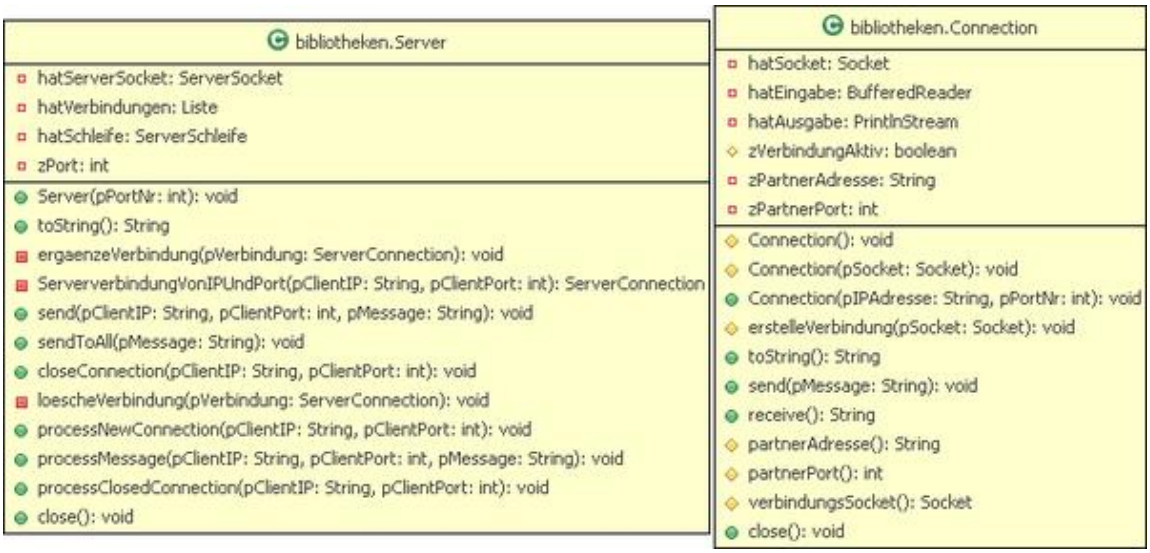


Abbildung 2 Klassendiagramm der Serverklassen des Zentralabiturs

Die Clientschnittstelle

Die Clientschnittstelle ist einfacher aufgebaut als die Serverschnittstelle. Sie muss lediglich über die Funktionalitäten „Senden einer Nachricht an den Server“, „Empfangen einer Nachricht“ und „Beenden des Sockets“ verfügen. Auf dem folgenden Klassendiagramm ist die Clientschnittstelle abgebildet:

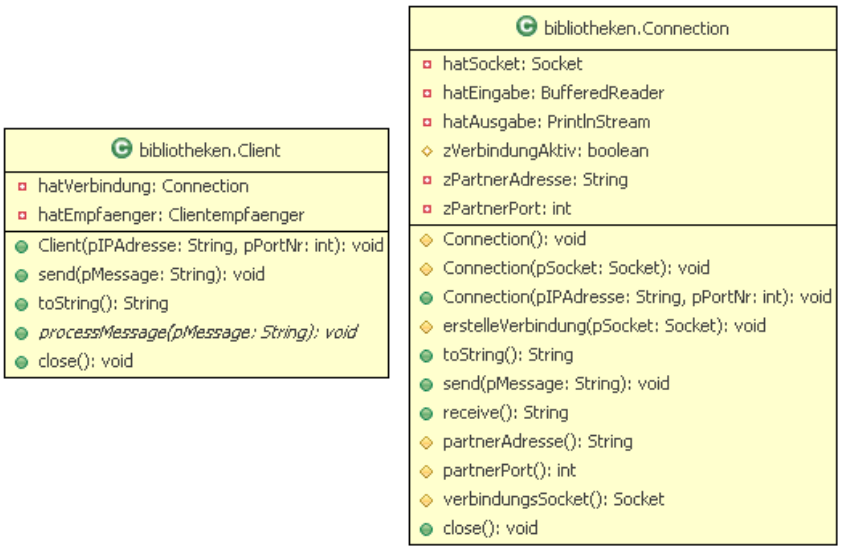


Abbildung 3 Klassendiagramm der Clientklassen des Zentralabiturs

3.2.3 Klassendiagramme des Servers

Analyse der Hauptklassen, „VoteServer“ und „VoteMain“

Die Klasse „VoteMain“ wird nach dem Ausführen der Klasse „Vote“ gestartet. In dieser Klasse wird ein neuer „VoteServerDatenDialog“ erzeugt, um die nötigen Daten wie Serverport, Serverpasswort, Servername sowie die Black- bzw. Whitelist abzufragen. Nach dem Bestätigen des Datendialogs wird ein Objekt der Klasse „VoteServer“ erzeugt. Dieses stammt direkt von der Zentralabiturklasse „Server“ ab und übernimmt alle Hauptfunktionalitäten des Servers, die vom Client in Anspruch genommen werden und von der GUI[5] aufgerufen werden. In der Klasse VoteServer wird u.a. auch die „VoteServerOberfläche“, die Hauptoberfläche, erzeugt. Auch die Klassen „BenutzerVerwaltung“ und „UmfragenVerwaltung“ sind hier als ein Attribut definiert.
Das folgende Klassendiagramm bildet die Hauptklassen schematisch ab (detailliertere Angaben siehe Gliederungspunkt 8.4.1):

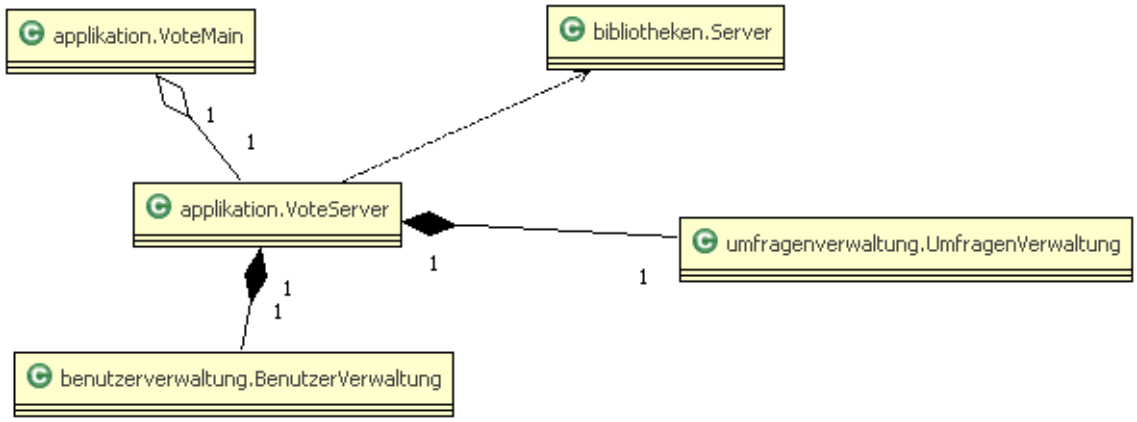


Abbildung 4 Schematisches Klassendiagramm der Klassen "VoteServer", "VoteMain", "Server", "UmfragenVerwaltung" und "BenutzerVerwaltung"

Die Klasse „UmfragenVerwaltung“

Die Klasse „UmfragenVerwaltung“ (siehe Abbildung 4) beinhaltet alle Methoden, um die Umfragen des Servers zu verwalten.

Die Klasse „BenutzerVerwaltung“

Die Klasse BenutzerVerwaltung (siehe Abbildung 4) bietet alle Funktionalitäten, um Benutzer zu verwalten sowie die Black- oder Whitelist zur gezielten Beschränkung der Nutzer.

Nebenklassen

Wichtige Nebenklassen des Servers sind die Klassen „VoteEinstellungen“, „VoteProtokoll“ sowie „VoteHilfsmittel“ (siehe Abbildung 5).

[5] GUI: „Graphical User Interface“

In der Klasse „VoteEinstellungen“ sind die benutzerspezifischen Einstellungen zu Funktionen des Servers gespeichert – zum Beispiel eine „Standard-Maximale-Teilnehmerzahl“. Nach dem Start des Vote-Servers wird geprüft, ob im Hauptverzeichnis[6] eine Konfigurationsdatei liegt und wenn dem so ist, werden die Konfigurationen aus dieser Datei geladen. Bei einer Änderung der Konfigurationen werden diese in der Konfigurationsdatei abgespeichert.

Die Klasse „VoteProtokoll“ enthält neben den Protokollbefehlen (siehe Gliederungspunkt 3.2.5) Informationen über die aktuelle Version des Vote-Servers und die XML-Tags zum Export der Umfrage.

In der Klasse „VoteHilfsmittel“ befinden sich statische Methoden, die nicht direkt zum Programminhalt gehören, aber wichtig zur Funktion des Vote-Servers sind – wie zum Beispiel das Bilden der MD5-Summe[7] von einer Zeichenkette.

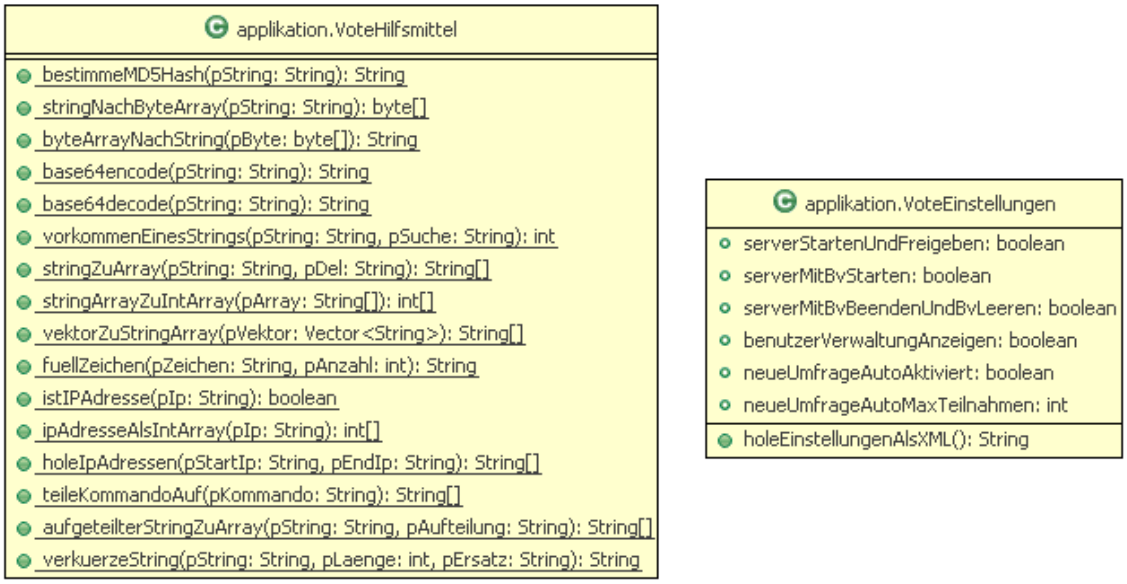


Abbildung 5 Klassendiagramm der Klassen „VoteEinstellugen“, „VoteHilfsmittel“

3.2.4 Klassendiagramme des Clients

Analyse der Hauptklassen

Die Klasse „VoteMain“ wird nach dem Ausführen der Klasse „Vote“ gestartet. In dieser Klasse wird ein neuer „VoteClientDatenDialog“ erzeugt, um die Serverdaten – wie IP-Adresse und Port des Servers – zu erfassen. Nach dem Bestätigen dieses Dialogs wird ein Objekt der Klasse „VoteClient“ erzeugt, das direkt von der Zentralabiturklasse „Client“ abstammt. In dieser Klasse befindet sich die Logik des Clients zum Empfangen von Umfragen und zur Teilnahme an Umfragen.

Das folgende Klassendiagramm bildet die Hauptklassen schematisch ab (detailliertere Angaben siehe Gliederungspunkt 8.4.2):

[6] Hauptverzeichnis: Verzeichnis in dem auch die ausführbare JAR-Datei liegt
[7] (Ullenboom, 2008)

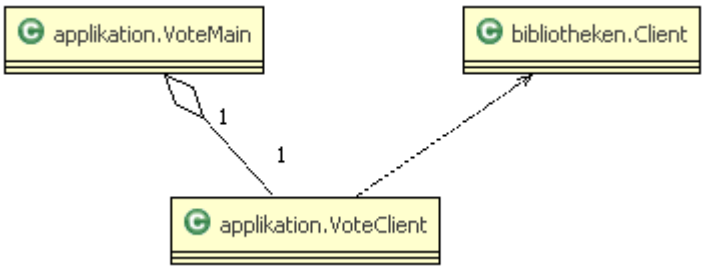


Abbildung 6 Klassendiagramm der Hauptklassen des Vote-Clients: „VoteMain“, „VoteClient“, „Client“

Nebenklassen

Wichtige Nebenklassen des Clients sind die Klassen „VoteProtokoll“ sowie „VoteHilfsmittel“ (siehe Abbildung 5), die mit den gleichnamigen Klassen des Vote-Servers identisch sind.

3.2.5 Das Protokoll

Wie unter 2.2.1 gesehen, sind Protokolle wichtig für die Kommunikation zwischen zwei Rechnern. Daher hat auch das System Vote ein eigenes Protokoll, das in der Klasse „VoteProtokoll“ untergebracht ist. Das Protokoll ist nach dem Vorbild des HTTP-Protokolls entwickelt (im Anhang ist die Protokollreferenz zu finden).

Die Kommunikation zwischen VoteServer und VoteClient findet über Kommandos statt, die Befehle beinhalten. Der Aufbau eines typischen Kommandos sieht wie folgt aus:

```
login; e8fa422bfb215053c69cd5a8d0262c7a; maximilianstrauch
```

Grundsätzlich teilt sich ein Kommando in drei Segmente auf: als Erstes wird der Befehl genannt, darauf folgen ein Semikolon und ein Leerzeichen. Das zweite Segment ist die MD5-Summe des Sendeinhalts, der als drittes Segment folgt, und wiederum durch ein Semikolon und ein Leerzeichen vom zweiten Segment abgetrennt ist. Falls das Kommando keinen Inhalt enthalten sollte, so fallen die Segmente zwei und drei weg.
Der Sinn der MD5-Summe liegt darin, prüfen zu können, ob ein Signalverlust vorhanden ist. Dies ermöglicht gegebenenfalls fehlerhaft übertragene Befehle zu ignorieren oder dies dem Server zu melden. Falls das dritte Segment mehrere Werte enthält, so sind diese durch Kommata von einander getrennt.

Eine wichtige Funktionalität des Protokolls ist neben der Anmeldung die Übertragung einer Umfrage. Dabei wird die Umfrage in ein XML-Dokument exportiert. Da diese XML-Dokumente sehr groß werden können (es kann bis zu „n“ Fragen in einer Umfrage geben), wird das XML-Dokument zeilenweise versendet:

```
...
single-survey; ...; <?xml version="1.0" encoding="UTF-8"?>
single-survey; ...; <umfrage id="0">
single-survey; ...; <title>Kunst</title>
single-survey; ...; </umfrage>
...
```

Dadurch ist es möglich auch längere Umfragen problemlos zu versenden. Vor dem Senden wird dem VoteClient mittels des Kommandos „single-surveys-comming-in“ mitgeteilt, wie viele Zeilen gesendet werden. Nach dem Senden der einzelnen Zeilen wird der VoteClient mit dem Kommando „single-survey-send“ darüber informiert, dass alle Zeilen an ihn gesendet wurden. Darauf werden diverse Methoden gestartet, die prüfen, ob die Übertragung erfolgreich war. Im Erfolgsfall wird die Umfrage zur Teilnahme auf dem Bildschirm angezeigt.

Zwischenfazit

Mit dem Abschluss der dargelegten Programmierschritte ist ein funktionales Umfragesystem entstanden, dass die gewünschten Mindestanforderungen zum Einsatz bei der in Abschnitt 3.1 dargelegten Zielgruppe erfüllen kann. Das System ist daher bereit, in einer Testphase evaluiert zu werden. Da das System nun in einer Beta[8]-Version vorliegt, wurde mit ausgewählten Testern ein Betatest durchgeführt, der im folgenden Abschnitt näher erläutert und evaluiert wird.

[8] Beta-Version: unfertige Version eines Computerprogramms; meistens sind Beta-Versionen die ersten Versionen eines Programms, die zum Testen von dem Hersteller veröffentlicht werden (Exil, 2009)

4. Evaluation

Im Rahmen des Informatikunterrichts ergab sich mit freundlicher Genehmigung des Fachlehrers – Herr Kaibel – Gelegenheit, die entwickelte Software mit 26 Personen zu testen. Die 26 Personen, die den VoteClient testeten, gehören zugleich auch der Zielgruppe an (siehe 3.1). Der VoteServer wurde zusätzlich von dem Fachlehrer getestet, der ebenso als Zielgruppe in Betracht kommt.

Die Evaluation erfolgte anhand eines Betatests, da VoteClient und VoteServer zu diesem Zeitpunkt den Beta-Status erreicht hatten[9].

4.1 Strukturierung des Betatests

Für eine differenzierte Auswertung des Betatests war die gewählte Strukturierung ausschlaggebend. Daher war ein differenziertes Testverfahren der einzelnen Module, wie Benutzerverwaltung mit zugehöriger Black- bzw. Whitelist, sowie ein differenzierter Rückmeldebogen erforderlich. So wurde der Betatest in drei Teile gegliedert, die auch die Gliederung der nachfolgenden Evaluation stellen: in die generelle Einschätzung, in die detaillierte Fehlerrückmeldung und in die Verbesserungsvorschläge. Kombiniert mit den vier Testphasen ergab sich somit ein differenziertes Beurteilungsschema.

Dem eigentlichen Betatest vorausgeschickt wurde eine kurze Präsentation zur Einführung der Testpersonen in die Grundlagen und Funktionen des VoteClient (siehe 8.3.2).

4.2 Generelle Einschätzung der Betatester

Generelle Einschätzung des VoteClient

Die Beurteilung des VoteClient durch die Betatester fiel insgesamt positiv aus mit wenigen negativen Einschätzungen. Leichte Schwächen sahen die Testpersonen in einzelnen Facetten.

In der Kategorie „Generell“ hat die Mehrheit der Tester Probleme bei der Beurteilung des Nutzens der Software (siehe Diagramm 1), da sie sich keine Anwendungsszenarien vorstellen können oder auch die Ehrlichkeit der Teilnehmer anzweifeln und daher die Verlässlichkeit der Umfrageergebnisse anzweifeln.

Dagegen beurteilen die Testpersonen die Fehlerberichte und die Softwarefunktionalitäten als „gut“ bis „sehr gut“ gelöst. Außerordentlich positiv wurde in dieser Kategorie die Einfachheit der Dateneingabe, also die Teilnahme an Umfragen, beurteilt. Mehr als Dreiviertel der Betatester beurteilten diesen Bereich mit „gut“ und „sehr gut“ - wie auch in Diagramm 1 erkennbar.

[9] Versionen: VoteServer, v0.2.5 beta, Build: 09.301; VoteClient, v0.2 beta, Build: 09.301

Der insgesamt mittelmäßig beurteilte generelle Eindruck ist auf die geringe Anzahl der Funktionen des VoteClient zurückzuführen.

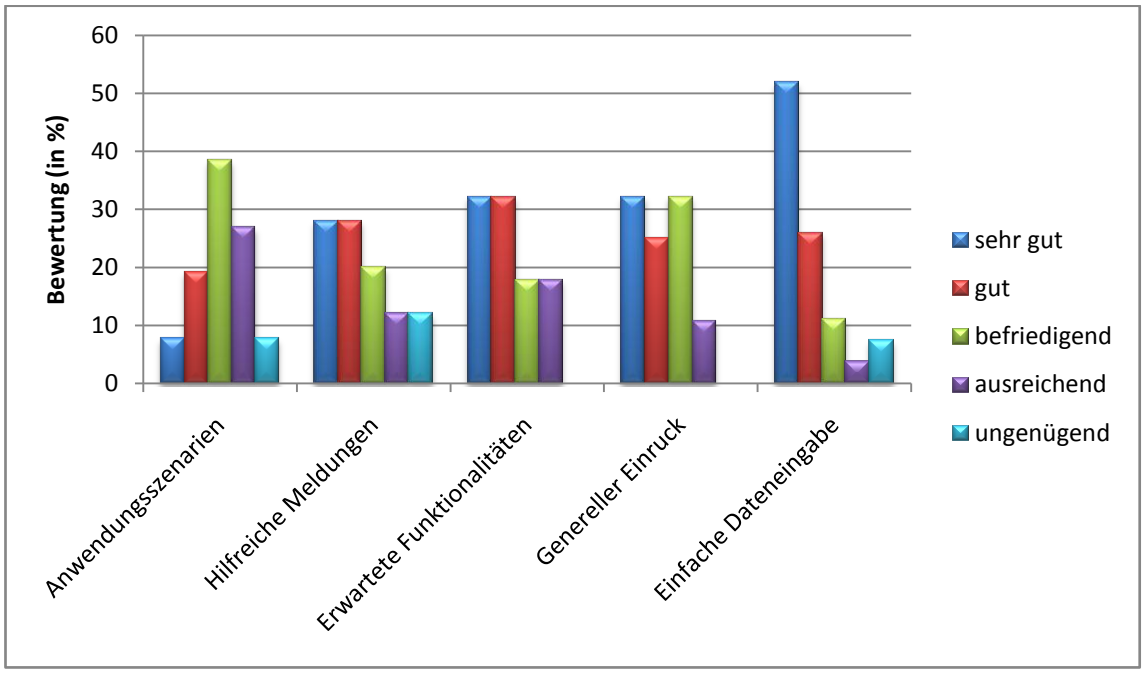


Diagramm 1 Kriterienbasierte Auswertung der Kategorie „Generell“

Im Bereich „Technischer Hintergrund/Durchführung von Operationen mit dem Programm“ ist die relative Mehrheit der Beurteilungen auf dem Niveau „gut“ bis „sehr gut“, deutlich an Diagramm 2 erkennbar. So sind die Ausgaben einfach zu verstehen, die Handhabung ist einfach und der logische Aufbau ist nachvollziehbar. Kritisiert wurde die gelegentlich fehlerhafte Ausführung von Operationen sowie das Fehlen einiger Funktionalitäten – auch erkennbar in Diagramm 2 durch die relativ hohen Anteile der Bewertungen „ungenügend“ und „ausreichend“.

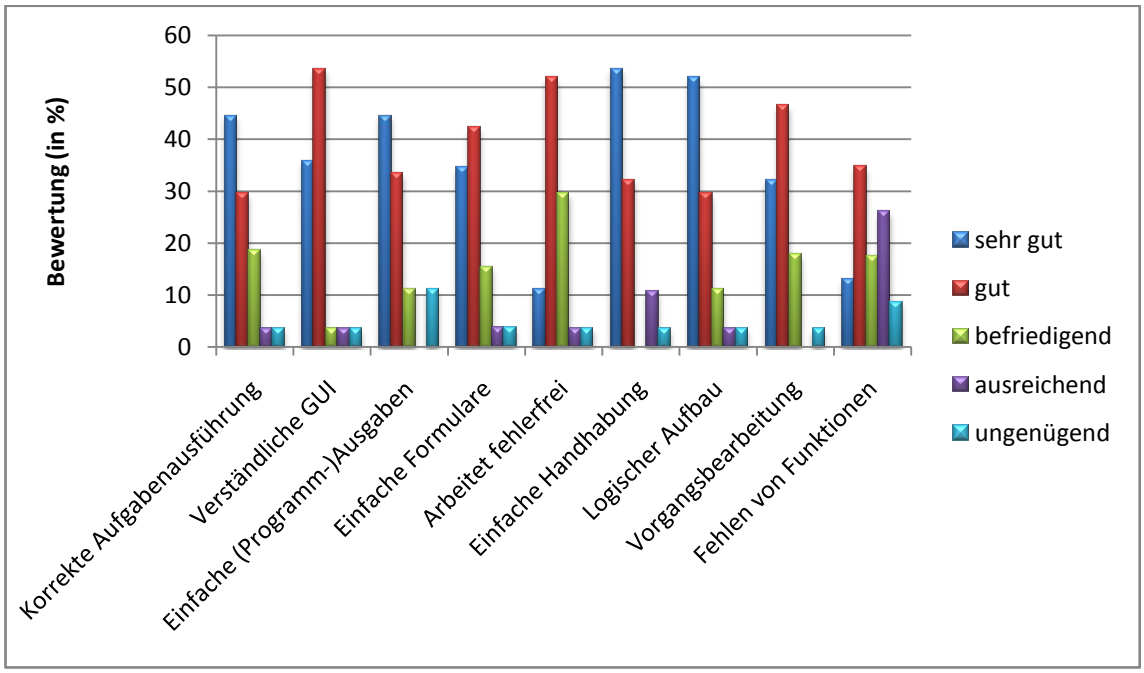


Diagramm 2 Kriterienbasierte Auswertung der Kategorie „Technischer Hintergrund/Durchführung von Operationen mit dem Programm“

In der Kategorie „Aussehen (Look and Feel)“ ist die Mehrheit der Beurteilungen auf einem positiven Niveau. Etwa Zweidrittel bis Dreiviertel der Betatester beurteilen die Oberfläche als professionell mit korrekter Rechtschreibung und einem „Corporate Design“, wie auch in Diagramm 3 erkennbar. Kritisiert werden hier lediglich die Piktogramme (oder auch Symbole) auf einigen Buttons, da diese den Benutzer in die Irre führen könnten.

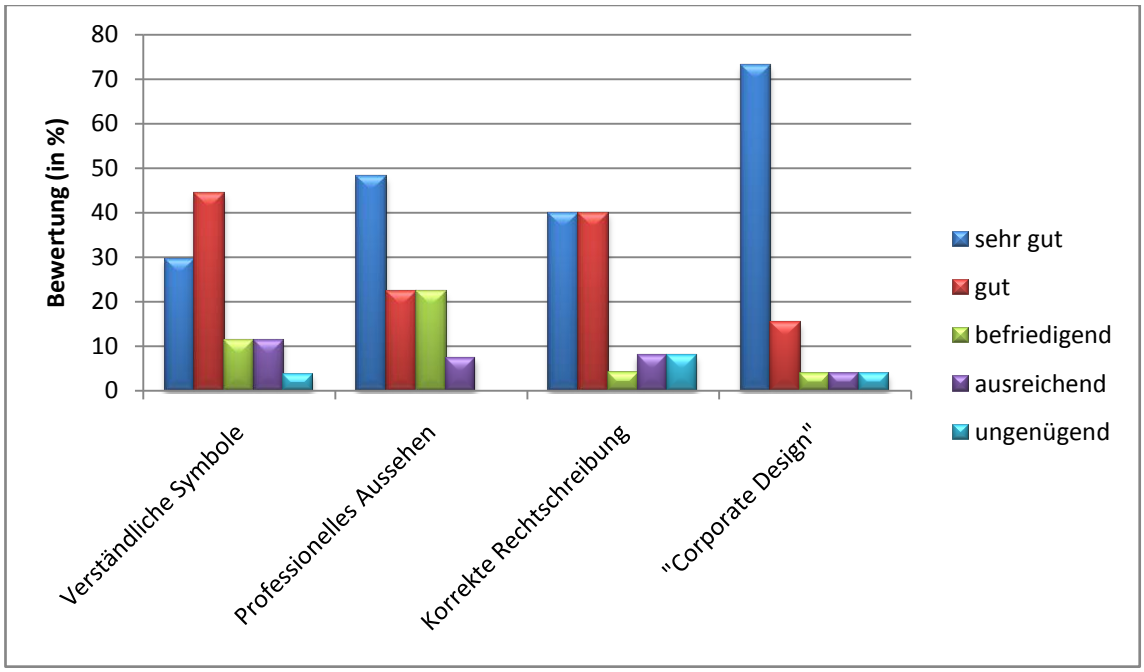


Diagramm 3 Krierienbasierte Auswertung der Kategorie "Aussehen (Look And Feel)"

Abschließend sollte darauf hingewiesen werden, dass an diesem Betatest Schüler in Arbeitsgruppen teilgenommen haben und somit eine Manipulation der Ergebnisse aufgrund eines gruppendynamischen Auswahlverhaltens nahe liegt. Daher kann der Betatest nur grobe Tendenzen liefern.

Generelle Einschätzung des VoteServer

Der VoteServer wurde durch Herrn Kaibel, als unabhängigem Tester, getestet. Somit ist auch eine Einschätzung des Vote-Servers aus Anwendersicht gewährleistet.

In der Kategorie „Generell“ schätzt der Betatester den VoteServer weitgehend als „gut“ bis „sehr gut“ ein. Kritisiert werden jedoch die optische Gestaltung und das Fehlen eines Kontextmenüs. Auch wird bemängelt, dass die Piktogramme der Buttons nicht aussagekräftig genug seien. Sehr positiv zu bewerten sei wiederum, dass keine Fehlermeldung während des Testlaufes zustande kam.

Die Kategorie „Technischer Hintergrund/Durchführung von Operationen mit dem Programm“ wurde durchschnittlich „gut“ bis „sehr gut“ bewertet. Auch an dieser Stelle wurde kritisiert, dass die Piktogramme der Buttons nicht eindeutig identifizierbar seien und dass die Übersichtlichkeit für Einsteiger in das Programm nicht immer gegeben sei.

In der Kategorie „Aussehen (Look and Feel)“ wird erneut auf die nicht eindeutige Identifizierbarkeit der Piktogramme verwiesen.

4.3 Fehlerauswertung und Rückschlüsse

Fehlerauswertung des VoteClient

Im Laufe des Betatests wurden acht unterschiedliche Fehler von den Betatestern gefunden, die während der Laufzeit des Vote-Clients auftraten.

Der am häufigsten aufgetretene Fehler war, dass nach der Aktualisierung der Umfragenliste keine Umfragen aufgelistet wurden, obgleich in der Statuszeile der Hinweis angezeigt wurde, dass „n“ Umfragen übertragen worden seien.
Dieser Fehler kann nicht mit der Überforderung des Servers zusammenhängen, da die Vote-Clients den Empfang von Elementen der Umfragenliste in den eigenen Protokollen festhielten und zudem stets die Zahl der aktiven Umfragen auf der Oberfläche dem Benutzer anzeigten.
Daher lässt sich schlussfolgern, dass die Umfragenliste nicht auf die GUI „gebracht“ wird, was nichts mit dem Netzwerkverkehr zutun haben kann, sondern mit der Methode, die in der Klasse VoteClient die Umfragenliste anzeigt.
In der Methode „verarbeiteServerAnfrage“ wird die Anfrage vom Server angenommen und geprüft. Da keine detaillierte Fehlerbeschreibung eines Benutzers vorliegt, ist nicht genau zu sagen wo der Fehler liegt. Es könnte also beispielsweise sein, dass eine Diskrepanz zwischen der Variable „zuErwartendeElemente“ und „ankommendeElemente“ besteht: beide beinhalten die Anzahl der Umfragen, die aufgelistet werden. Es könnte aber auch ein Fehler in der Methode „neueUmfrage“ vorliegen. Diese Methode zeigt die Umfrage auf der Oberfläche an.
Der Schwachpunkt dieser Funktion liegt vermutlich darin, dass die Elemente der Umfragenliste einzeln verarbeitet werden. Da es bei der Umfrageübertragung – die ja im XML-Format statt findet – keine Fehler gab, sollte zur Optimierung das XML-Format auch für die Umfragenliste eingeführt werden.

Der zweit häufigste Fehler bestand darin, dass Umfragen zunächst ganz unten am Fensterrand aufgelistet werden und darunter viele leere Umfragen (auf der Liste kann man weder Icon noch Umfragetitel erkennen) sind. Dies hängt vermutlich mit dem ersten Fehler zusammen. Er lässt sich wohl auch auf den Schwachpunkt der Übertragung der Umfragenliste zurückführen.

Am dritt häufigsten trat der Fehler „java.lang.IllegalStateException: Timer already cancelled“ auf. Die Statusanzeige hat einen Timer, der dafür sorgt, dass der angezeigte Hinweis nach zehn

Sekunden entfernt wird. Ursächlich wird hier eine bestimmte Methodenkonstellation sein, dahingehend einen beendeten Timer zu beenden. Es lässt also auf einen logischer Fehler in einer der Methoden, die sich um die Statusanzeige sorgen, schließen. Dieser kann durch erneutes Prüfen der Logik dieser Methoden eliminiert werden.

Neben den aufgeführten Fehlern wurden die Umlaute nicht richtig angezeigt. Manchen Testern war trotz Freischaltung – die betreffenden Benutzer standen auf der Whitelist – die Anmeldung zeitweilig nicht möglich. Auch wurde in einigen Fällen die Oberfläche des VoteClient nicht angezeigt. Diese kleineren Fehler lassen sich durch Modifikationen der zuständigen Klassen beheben. Zum Beispiel kann ein „Umlautkonvertor“, der die Umlaute für die Übertragung verändert, eingefügt werden, oder ein erneuter Test der Whitelist mit Namen, die Sonderzeichen enthalten, kann etabliert werden.

Fehlerauswertung des VoteServer

Bis auf eine Fehlfunktion der Benutzerliste ist bei dem Betatest des Vote-Servers keine weitere Fehlfunktion aufgetreten. Von Zeit zu Zeit verschwand die Benutzerliste, wurde dann aber wieder angezeigt. Eine direkte Erklärung hierfür gibt es nicht, jedoch könnte es möglich sein, dass bei der Aktualisierung der Benutzerliste folgender Fehler aufgetreten ist: die Benutzerliste wurde geleert und die Methode zur Auflistung aller aktiven Benutzer wurde aus nicht geklärten Gründen nicht aufgerufen.

Während des Testlaufes des Vote-Servers durch den unabhängigen Betatester fielen keine Programmfehler auf.

5. Ausblick

5.1 Zukünftige Erweiterungen des VoteClient

Im Rahmen des Betatests wurden auch Verbesserungsvorschläge der Betatester erfasst. Die Verbesserungsvorschläge aller Betatester bezogen sich im Wesentlichen auf die gleichen Gegenstände.

Die Mehrzahl der Betatester möchte die Umfrageergebnisse auch mit dem VoteClient betrachten, obgleich dies die Gefahr der Beeinflussung des Benutzers durch bereits vorliegende Umfrageergebnisse eröffnet. Um diese Gefahr einzudämmen wird vorgeschlagen, dass man die vorliegenden Umfrageergebnisse erst dann betrachten kann, wenn die Teilnahme an der Umfrage abgeschlossen ist. Einige Tester wünschen sich die Angabe der absoluten Anzahl der Teilnehmer an einer Umfrage, sodass die Auswertungen in relativen Zahlen aussagekräftiger werden. Zusätzlich sollten die relativen Zahlen mit Nachkommastellen angegeben werden, um so die Bewertungsgenauigkeit zu erhöhen.

Ein Teil der Betatester wünscht sich klarere Meldungen. Wenn ein Benutzer vom Administrator entfernt wird, so soll dieser darüber informiert werden. Außerdem soll ein genauer Hinweis erscheinen, wenn ein User beispielsweise auf der Blacklist steht und sich nicht anmelden kann. Weiteren Wünschen zufolge sollte nach einer Umfrageteilnahme ein Fenster erscheinen, in dem für die Umfrageteilnahme gedankt wird.

Ein Betatester äußerte den Vorschlag, die Eingabe der Verbindungsdaten (Server-IP-Adresse und Serverport) als Menüpunkt im Hauptmenü zu erstellen und durch eine Konfigurationsdatei zu speichern. Somit könnte für künftige Sitzungen der lästige Anmeldedialog umgangen werden. Dies würde eine enorme Vereinfachung und Verkürzung der Anmeldeprozedur mit sich bringen und wäre somit sehr positiv für den Benutzer des Vote-Clients.

Viele Benutzer regen an, die Piktogramme der Buttons klarer zu gestalten, zu jeder Umfrage ein Bild bereitzustellen sowie das Passwort im Passwortfeld für das Serverpasswort nicht im Klartext anzuzeigen, sondern durch Echozeichen.

5.2 Zukünftige Erweiterungen des VoteServer

Aus der Sicht des Entwicklers ist der VoteServer weitgehend ausgereift, kann aber dennoch verbessert werden. So könnte es eine Funktion zum Zurücksetzen einer Umfrage geben, um die Umfrage in den Normalzustand zu versetzten, damit wieder Benutzer daran teilnehmen können.
Im Rahmen des Aussehens und der GUI könnte man einen Button zur Aktualisierung der Benutzerliste einfügen sowie eine Funktion, um das Umfrageergebnis in einem großen Fenster anzuzeigen.

Erweiterungsvorschläge des unabhängigen Testers

Aus inhaltlicher Sicht kann der VoteServer um die Angabe der Teilnahmeanzahl an einer Umfrage erweitert werden. Diverse Funktionalitäten könnten aufgenommen werden, um ein Umfrageergebnis zusätzlich als Diagramm darzustellen.
Die Komplexität des Vote-Servers sollte durch „Wizards“[10] sowie Kontextmenüs aufgebrochen werden, sodass eine intuitive Benutzung vereinfacht wird. Außerdem sollten die Attribute einer Umfrage (maximale Teilnehmeranzahl etc.) im Informationsfenster durch einen Doppelklick editierbar sein.
Sofern die gewählten Piktogramme der Buttons in der als überwiegend unklar empfundenen Darstellungsweise belassen werden, sollten die Buttons um erläuternde Schriftzüge erweitert werden. Die Buttons, mit denen beispielsweise Umfragen erstellt werden, sollten unter den Dialog gesetzt werden, um die natürliche Lesereihenfolge zu beachten.

Fazit

Nach Erstellung eines anwendungsreifen Systems und Evaluation im Praxistest durch die Zielgruppe kann mit dieser wissenschaftlichen Arbeit die Entwicklung eines Umfragesystems vorgelegt werden.
Der Betatest zeigt die praktische Einsatzfähigkeit des Systems und die Verbesserungsvorschläge tragen zur Optimierung im Zuge einer gezielten Weiterentwicklung bei.
Es ist damit gelungen, eine funktionale Software für ein Umfragesystem in lokalen Netzwerken, wie z.B. Schulen, zu entwickeln.

[10] „Wizards“: Dialoge, die Schritt für Schritt z.B. eine Umfrage erstellen und die möglichen Optionen als Fragen formuliert an den Benutzer stellen und somit alle nötigen Daten sammeln

6. Literaturverzeichnis

Exil. 2009. Entwicklungsstadium (Software). [Online] Wikipedia, 12. März 2009. [Zitat vom: 12. März 2009.] http://de.wikipedia.org/wiki/Entwicklungsstadium_(Software).

F.A. Brockhaus GmbH. 1999. *Der Brockhaus in fünfzehn Bänden.* Leipzig-Mannheim : F.A. Brockhaus GmbH, 1999.

Münz, Stefan und Nefzger, Wolfgang. 2005. *HTML Handbuch (Studienausgabe).* Poing : Franzis Verlag GmbH, 2005.

Ullenboom, Christian. 2008. Galileo Computing :: Java ist auch eine Insel (8. Auflage). [Online] Galileo Computing, 16. Dezember 2008. [Zitat vom: 24. März 2009.] http://openbook.galileocomputing.de/javainsel8/javainsel_04_008.htm#mj9430a7cb01468b2599ae42e412071dfd. 978-3-8362-1371-4.

7. Erklärung der Selbstständigkeit

Ich erkläre, dass ich die Facharbeit ohne fremde Hilfe angefertigt habe und nur die im Literaturverzeichnis angeführten Quellen und Hilfsmittel verwendet habe. Alle Zitate und Übernahmen sind im Text der Facharbeit kenntlich gemacht.

Maximilian Strauch

8. Anhang

8.1 Diverse Verzeichnisse

8.1.1 Abbildungsverzeichnis

8.1.2 Diagrammverzeichnis

8.2 Vote-Protokoll-Referenz

8.2.1 Serverbefehle

Befehl	Inhalt	Beschreibung
login-sucsessful	-	Anmeldung am Server erfolgreich
user-message	[Nachricht]	Nachricht an einen Benutzer
survey-list-item	[Anzahl der Umfragen], [Umfragen-Id], [Umfragen-titel]	Überträgt ein Element (also eine Umfrage) der Umfrageliste an einen Client
term	-	Beendet einen Benutzer
survey-send	-	Umfrageliste wurde gesendet
single-surveys-comming-in	[Anzahl der Zeilen des XML-Dokuments]	Neue Umfrage wird übertragen
single-survey	[Zeile einer Umfrage]	Eine Zeile der zu übertragenden Umfrage
single-survey-send	-	Umfrage wurde übertragen
new-survey-available	-	Neue Umfragen liegen bereit
survey-successful	-	Umfrageteilnahme war erfolgreich

8.2.2 Clientbefehle

Befehl	Inhalt	Beschreibung
login	[Name], [Passwort]	Meldet einen Benutzer am Server an
get-survey-list	-	Fordert die Umfrageliste an
get-survey	[Umfragen-Id]	Fordert eine Umfrage an
survey-result	[Umfragen-Id], [Umfragen-teilnahme-String]	Überträgt das Ergebnis einer Umfrage

8.2.3 Fehlerbefehle

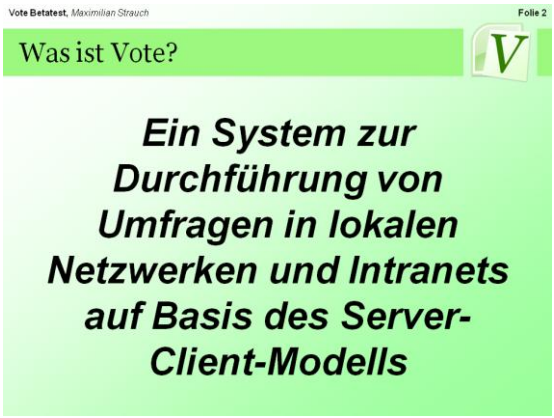
Sender	Befehl	Inhalt	Beschreibung
Client	survey-cant-read	-	Umfrage kann nicht gelesen/geparst werden
Server	login-passwd-missing	-	Passwort fehlt
Server	login-passwd-wrong	-	Passwort ist fehlerhaft
Server	survey-not-available	-	Umfrage ist nicht erreichbar
Server	survey-not-successful	-	Umfrageteilnahme war nicht erfolgreich
Server	survey-no-longer-available	-	Umfrage ist nicht mehr (länger) erreichbar
Server	server-not-reachable	-	Server ist nicht erreichbar
Server	bv-not-reachable	-	Benutzerverwaltung ist nicht erreichbar
Server	bv-bp-error	-	Benutzerpuffer ist fehlerhaft
Server/ Client	command-error	-	Kommando ist nicht bekannt
Server/ Client	arguments-error	-	Kommandoargumente sind fehlerhaft
Server/ Client	unknown-user	-	Benutzer ist unbekannt
Server/ Client	argument-read-error	-	Kommandoargumente können nicht gelesen werden

8.3 Betatestauswertung

8.3.1 Zeitplan des Betatest des Vote-Projekts (für ca. 45 Minuten)

Phase	Vorgehen	Zeitbedarf
-	Präsentation des Projekts; Einführung des Auditoriums und Ausgabe der Evaluationsunterlagen	10 Min.
I	**Test der Benutzerverwaltung (Blacklist)** *Teil 1:* Test mit Blacklist und der Option „nur Namen“ *Teil 2:* Test mit Blacklist und der Option „nur IP“ *Teil 3:* Test mit Blacklist und den Optionen „nur Namen“ und „nur IP“	5 Min. *(Jeweils nur Anmelden, 1 Umfrage, Abmelden)*
II	**Test der Benutzerverwaltung (Whitelist)** *Teil 1:* Test mit Blacklist und der Option „nur Namen“ *Teil 2:* Test mit Blacklist und der Option „nur IP“ *Teil 3:* Test mit Blacklist und den Optionen „nur Namen“ und „nur IP“	5 Min. *(Jeweils nur Anmelden, 1 Umfrage, Abmelden)*
III	**Test des gesamten Projektes mit allen Funktionen,** mit dem Entwickler *(Kicken, Nachricht, Umfragenstatus etc.)*	15 Min.
IV	**Test des gesamten Projektes mit allen Funktionen,** durch Herrn Kaibel (somit kann auch „VoteServer“ evaluiert werden) *(Kicken, Nachricht, Umfragenstatus etc.)*	5 Min.
-	Rückgabe der Evaluationsbögen	5 Min.

8.3.2 Einführung der Tester in Vote (Präsentation)



8.3.3 Evaluationsbogen mit Stimmauswertung der Betatester

Stimmauswertung des Betatests vom 02.Februar 2009 mit 26 + 1 Betatestern; Versionen: VoteServer, v0.2.5 beta, Build: 09.301; VoteClient, v0.2 beta, Build: 09.301.

Genereller Evaluationsbogen (mit Stimmauswertung)

Kriterien	**Bewertung** **(1= ungenügend, 5= sehr gut)**					**Sonstige Kommentare**
	1	2	3	4	5	
Generell						
Diese Software ist jetzt oder wird in Zukunft für mich/uns nützlich sein	2	7	10	5	2	• z.B. Schülerzeitung • Ehrlichkeit der Teilnehmer zweifelhaft • „Umfragen sind immer praktisch“
Die Fehlerberichte/Meldungen sind hilfreich	3	3	5	7	7	• Keine erhalten • hilfreich
Die Software hat die nötigen Funktionen, die ich/wir erwarte/n	0	5	5	9	9	• Teilnahme/Erfolgsrückmeldung • Mehrfachteilnahme muss einsichtig sein
Genereller Eindruck von Vote (Warum?)	0	3	9	7	9	• spannend • eher langweilig • nicht umfangreich • gute Idee • schöne Oberfläche • „sehr schön“
Dateneingabe (Teilnahme an Umfragen) ist einfach gestaltet	2	1	3	7	14	• Doppelklick/Tastatur mit einbeziehen • „nur ein Paar Klicks nötig“ • sehr übersichtlich

Technischer Hintergrund/Durchführung von Operationen mit dem Programm						
Führt gegebene Aufgaben richtig aus	1	1	5	8	12	• mehrfache Umfrageteilnahme möglich • keine Umfragenteilnahme möglich
Toolbars, Befehle und Optionen/Dialoge sind verständlich	1	1	1	15	10	• komplizierter Anmeldeprozess
Ausgaben sind einfach zu verstehen (wenn nicht, welche?)	3	0	3	9	12	• Fehlermeldungen z.T.
Formulare sind einfach/nützlich	1	1	4	11	9	• zu klein
Fehlerfreie Durchführung von Operationen	1	1	8	14	3	• Fehler bei der Aktualisierung • Fehler bei der Anmeldung
Einfache Handhabung des Programms	1	3	0	9	15	• IP-Adresse kennt evtl. nicht jeder
Logischer Aufbau der Handlungsabläufe	1	1	3	8	14	• Fehlerhafte Piktogramme auf den Buttons
Software bearbeitet Vorgänge wie erwartet	1	0	5	13	9	
Fehlen bestimmte Funktionen (wenn ja, welche?)	2	6	4	8	3	• Ergebnisansicht • Ansicht der Gesamtzahl der Teilnahmenanzahl
Aussehen (Look and Feel)						
Farben, Symbole und Grafiken sind einfach und verständlich	1	3	3	12	8	• Aus dem Pfeil einen Haken machen • schöne, große Symbole
Sieht professionell aus	0	2	6	6	13	• Buttons „besser“ einbinden • sehr einfach gestaltet • ä, ö und ü fehlerhaft in der Ausgabe
Rechtschreibung/Grammatik bei (System-)Ausgaben korrekt	2	2	1	10	10	
Die verschiedenen Fenster haben ein gemeinsames Aussehen (Corporate Design) bzw. einen einheitlichen Aufbau	1	1	1	4	19	

Fehlerauswertung (mit Stimmauswertung)

Nr.	Systemausgabe/Fehler	Ursache(n)	Anzahl
1	Alle Umfragen empfangen, es werden aber keine angezeigt.	• evtl. dass sich alle gleichzeitig angemeldet haben • bei zweitem Laden/Anmelden tritt der Fehler nicht mehr auf	7
2	Es werden viele Umfragen ohne Titel aufgelistet; Anderen ganz unten	-	4
3	„java.lang.IllegalStateException: Timer already cancelled“; nach Umfragenteilnahme verschwindet das Fenster nicht	-	3
4	„An dieser Umfrage können Sie nicht mehr teilnehmen“	-	3
5	Vote-Client geladen (Systemausgabe); wird aber nicht angezeigt	-	1
6	Umlaute (ä, ö, ü) funktionieren nicht	-	1
7	Trotz Freischaltung Anmeldung nicht möglich (nur manchmal)	-	1
8	Mehrfache Umfrageteilnahme möglich	-	1

Verbesserungsvorschläge der Betatester zum VoteClient

- Aussehen („Look and Feel“):
  - Pfeil durch einen Haken ersetzen
  - Zu jeder Fragen/Umfrage ein Bildchen
  - Hinweis folgender Art: „Danke für Ihre Teilnahme.“
  - Passwortfeld ist nicht „verschlüsselt“
- Inhalt/Programmfunktionen
  - Umfrageergebnisse auch für Nutzer sichtbar (nach Umfrageteilnahme); Problematisch: Nutzerbeeinflussung
  - Gesamtzahl der Teilnahmen angeben (zur absoluten Auswertung sinnvoll)
    - Nachkommastellen anzeigen
  - IP/Port sollte als Menüpunkt erstellt werden und die Verbindungsdaten in einer Konfigurationsdatei gespeichert werden
  - Bei dem Entfernen von Umfragen durch den Admin sollte eine Meldung erscheinen

- genauer Hinweis, an die Nutzer, welche auf der Blacklist stehen
- Umlaute sollten richtig angezeigt werden
- Aktualisieren-Button (Umfragen)

Verbesserungsvorschläge zum VoteServer

- Umfrage „leeren“-Button: so wird aus einer „benutzten“ Umfrage wieder eine „neue“ Umfrage
- Benutzeranzeigen-Aktualisierungsbutton
- Umfragenauswertungsfenster in Groß (Großer Dialog?)
- Anzahl der Teilnehmer ausgeben
- In „Informationen“ sollte man Einträge editieren können, z.B. „Umfragestatus“ und „Max. Teilnahme“ durch Doppelklick
- Kontextmenü für Benutzer und Umfragen
- Buttons mit Schrift wären manchmal hilfreich; z.B. der grüne Pfeil ist nicht selbsterklärend
- Button für „Umfrage erstellen“ und „Abbrechen“ unter die Dialoge; siehe Lesereihenfolge
- (Umfragen-)Ergebnis als Diagramm darstellen

8.4 „JavaDoc“ Klassenübersicht

8.4.1 Hauptklassen des Vote-Servers

VoteMain.java (applikation.VoteMain)

Constructor Summary

`VoteMain(boolean pDebug)`	**Konstruktor - erzeugt eine neue VoteMain-Instanz**

Method Summary

void	**beendeVoteServer**() beendet die gesamte Applikation
boolean	**debugModusAktiv**() true, wenn der Debugmodus aktiv ist
void	**serverDatenErfassen**(java.lang.String pSName, int pSPort, java.lang.String pSPwd, java.lang.String pSMsg, boolean pWhiteList, boolean pUseIp, boolean pUseName, java.lang.String[] pIpListe, java.lang.String[] pNamensListe) nimmt die erfassten Daten an und startet die Oberflaeche
void	**starteVsdd**() startet den VoteServerDatenDialog

VoteServer.java (applikation.VoteServer)

Constructor Summary

`VoteServer(VoteMain pVoteMain, java.lang.String pSName, int pSPort, java.lang.String pSPwd, java.lang.String pSMsg, boolean pWhiteList, boolean pUseIp, boolean pUseName, java.lang.String[] pIpListe, java.lang.String[] pNamensListe, boolean usePasswd, boolean pDebug, VoteEinstellungen pVe)`	**Konstruktor - erzeugt einen neuen VoteServer**

Method Summary

void	**aktualisereListe**(boolean pWhiteList, boolean pUseIp, boolean pUseName, java.lang.String[] pIpListe, java.lang.String[] pNamensListe) aktualisiert die Benutzerverwaltungsliste
void	**aktualisiereEinstellungen**(VoteEinstellungen pVeTemp) aktualisiert die (Benutzer-)Einstellungen
Benutzer[]	**alleBenutzer**() gibt alle Benutzer zurueck
void	**ausgabeBenutzerVerwaltung**() gibt den Inhalt der Benutzerverwaltung auf der Konsole aus
void	**ausgabeUmfragenVerwaltung**() gibt den Inhalt der Umfragenverwaltung auf der Konsole aus
void	**beendeVoteServer**() beendet den VoteServer
BenutzerVerwaltung	**benutzerVerwaltung**() gibt das Benutzerverwaltungsobjekt zurueck
void	**benutzerVerwaltungOnline**(boolean pStatus)

	setzt die Benutzerverwaltung online/offline
boolean	**debugModusAktiv**()
	gibt zurueck, ob der Debugmodus aktiv ist
void	**entferneAusBenutzerPuffer**(java.lang.String pClientIP, int pClientPort)
	entfernt einen Clienten aus dem Benutzerpuffer
void	**entferneClient**(java.lang.String pClientIp, int pClientPort)
	entfernt einen Clienten aus der Verwaltung
void	**entferneServerPasswort**()
	entfernt das Serverpasswort
boolean	**entferneUmfrage**(int pUmfragenId)
	entfernt eine Umfrage
void	**exportiereUmfrage**(int pUmfragenId, int pModus, java.io.File pDatei, java.lang.String pExtension)
	exportiert eine Umfrage
void	**frageDatenAb**(int mode, java.lang.String pClientIP, int pClientPort)
	fragt Daten vom Clienten ab
void	**fuegeFrageZuUmfrage**(int pUmfragenId, java.lang.String pTitle, java.lang.String[] pAntworten, boolean pSingleChoice)
	fuegt eine Frage zu einer Umfrage
int	**holeAusBenutzerPuffer**(java.lang.String pClientIP, int pClientPort)
	holt einen Clienten aus dem Benutzerpuffer
String[][]	**holeServerInfos**()
	gibt die Serverinformationen zurueck
Umfrage	**holeUmfrageMitId**(int pUmfragenId)
	holt eine Umfrage anhand der Id
Umfrage	**importiereUmfrage**(java.lang.String pPath)
	importiert eine Umfrage
void	**kickeAlleNutzer**()
	entfernt alle Benutzer
void	**kickeBenutzer**(java.lang.String pIp, int pPort)
	entfernt einen Benutzer
void	**neuerKonsolenEintrag**(java.lang.String pEintrag, int pModus)
	erstellt einen neuen Konsoleneintrag auf der Serverkonsole Modi: 0 Umfragenverwaltung 1 Benutzerverwaltung 998 Speziell fuer Fehler 999 Speziell fuer Erfolge
int	**neueUmfrage**(java.lang.String pTitel, int pMaxTeilnahme)
	erzeugt eine neue Umfrage
void	**processClosedConnection**(java.lang.String pClientIP, int pClientPort)
	verarbeitet eine geschlossene Verbindung
void	**processMessage**(java.lang.String pClientIP, int pClientPort, java.lang.String pNachricht)
	verarbeitet eine neue Nachricht
void	**processNewConnection**(java.lang.String pClientIP, int pClientPort)
	verarbeitet eine neue Verbindung
void	**sendeNachricht**(java.lang.String pNachricht, java.lang.String pIp, int pPort)
	sendet eine Nachricht an einen Benutzer
java.lang.String	**serverNachricht**()

	gibt die aktuelle Servernachricht zurueck
void	**serverOnline**(boolean pStatus) de/aktiviert den Server
void	**setzeServerNachricht**(java.lang.String pNachricht) setzt eine neue Servernachricht
void	**setzeServerPasswort**(java.lang.String pPasswort) setzt ein neues Serverpasswort
void	**setzeUmfrageAktiv**(int pUmfragenId, boolean pAktiv) setzt eine Umfrage aktiv oder inaktiv
void	**starteBenutzeroberflaeche**() startet die grafische Oberflaeche
boolean	**umfrageAktiv**(int pUmfragenId) gibt den Umfragenstatus zurueck
void	**umfragenAenderungAnClient**() sendet an alle den Hinweis, dass neue Umfragen bereit stehen

8.4.2 Hauptklassen des Vote-Clients

VoteMain.java (applikation.VoteMain)

Constructor Summary

VoteMain**(boolean pDebug)** **Konstruktor - erzeut ein neues VoteMain-Objekt**

Method Summary

void	**beendeClient**() beendet den kompletten VoteClienten
void	**datenEingegeben**(java.lang.String pIp, int pPort) Erzeugt einen neuen VoteClient und versucht ein Socket aufzubauen
boolean	**debugModusAktiv**() gibt zurueck, ob der Debugmodus aktiv ist
void	**init**() erzeugt einen neuen Datendialog

VoteClient.java (applikation.VoteClient)

Nested Class Summary

class	**VoteClient.VoteServerAntwortTimer** VoteServerAntwortTimer

Constructor Summary

VoteClient**(VoteMain pVm, java.lang.String pIp, int pPort, java.lang.String pBenutzerName)** **Konstruktor - erstellt einen neuen VoteClient**

Method Summary

void	**anUmfrageTeilgenommen**(int pUmfragenId, java.lang.String pUmfragenTeilnahme) sendet eine Umfragenteilnahme an den Server
void	**beendeClient**() beendet den VoteClienten

void	**beendeKeineAntwortVomServerTimer**() hebt den 'KeineAntwortVomServer'-Timer auf
boolean	**debugModusAktiv**() gibt zurueck, ob der Debugmodus aktiv ist
void	**frageUmfrageAb**(int pUmfragenId) fragt eine Umfrage am Server ab
void	**init**() Initialisiert einen neuen VoteClienten und sendet die Anmeldenanfrage
void	**keineAntwortVomServer**() wird aufgerufen, wenn vom Server keine Antwort kommt
void	**processMessage**(java.lang.String pNachricht) bereitet eine Serveranfrage vor
void	**starteKeineAntwortVomServerTimer**() erzeugt einenen neuen 'KeineAntwortVomServer'-Timer
void	**starteOberflaeche**() startet die Clientoberflaeche
void	**umfrageNichtErhalten**() Umfrage nicht erhalten; Fehlermeldungsausgabe