Erstellt von: Jacqueline Kaefer & Luca Katzenberger

Matrikelnummer: XXXXXXX & 2577508

Betreut von: Prof. Dr. Michael Bauer

Datum: 01.03.2019

Chatprogramm

Dokumentation

Inhaltsverzeichnis

[I. Lastenheft 2](#_Toc1066738)

[1. Einleitung 2](#_Toc1066739)

[2. Konzept und Rahmenbedingungen 2](#_Toc1066740)

[3. Anforderungen 3](#_Toc1066741)

[4. Erweiterungen 3](#_Toc1066742)

[5. Genehmigung 3](#_Toc1066743)

[6. Anhang 3](#_Toc1066744)

[II. Pflichtenheft 4](#_Toc1066745)

[III. Theoretischer Hintergrund 5](#_Toc1066746)

[1. IP-Adressen und Ports 5](#_Toc1066747)

[2. UDP-Protokoll 5](#_Toc1066748)

[IV. Technische Dokumentation 6](#_Toc1066749)

[1. Die grundlegenden Funktionen des Programms 6](#_Toc1066750)

[2. Weitere Funktionen des Programms 9](#_Toc1066751)

[3. Die Oberfläche 10](#_Toc1066752)

[V. Anwenderdokumentation 12](#_Toc1066753)

[1. Starten des Chatprogramms 12](#_Toc1066754)

[2. Kontakte hinzufügen und löschen 12](#_Toc1066755)

[3. Nachrichten Senden 12](#_Toc1066756)

[4. Nachrichten empfangen 12](#_Toc1066757)

[5. Speicherung 13](#_Toc1066758)

[6. Fehlerbehebung 13](#_Toc1066759)

# Lastenheft

## Einleitung

Dieses Lastenheft beschreibt die Anforderungen für ein zu entwickelndes Chatprogramm. Dieses Programm wird im Rahmen des Moduls Softwareentwicklung als Informatikprojekt umgesetzt.

### Abkürzungen

* UDP-Protokoll: User Datagram Protokoll
* IP-Adresse: Internet Protokoll Adresse

### Projektteam

|  |  |  |
| --- | --- | --- |
| Rolle / Rollen | Name | E-Mail |
| Auftraggeber | Herr Prof. Dr. Michael Bauer |  |
| Projektteammitglied | Jacqueline Kaefer |  |
| Projektteammitglied | Luca Katzenberger | luca.katzenberger@hotmail.de |

## Konzept und Rahmenbedingungen

### Nutzen des Anwenders

Der Anwender kann Textnachrichten mit anderen Kontakten auszutauschen. Dabei hat er die Möglichkeit mit mehreren Teilnehmern parallel zu kommunizieren und zwischen den Chats zu wechseln.

Der Vorteil gegenüber herkömmlicher Kommunikation per Email ist, dass aus dem Chatten ein wesentlich schnelleres Antworten resultiert. Dies liegt daran, dass Chatnachrichten einen eher formlosen Charakter haben. Dadurch steigt die Kommunikationsgeschwindigkeit.

### Benutzer / Zielgruppe

Die Anwender dieses Programms sind Mitarbeiters eines Betriebs, da dieses Programm als firmeninternes Chatprogramm zur schnellen Alternative für Emails verwendet werden soll.

Darüber hinaus kann es auch im privaten Rahmen zur Kommunikation genutzt werden.

### Systemvoraussetzungen

Für die Verwendung des Programms wird ein Windowsrechner vorausgesetzt, auf dem .NET Anwendungen verwendet werden können. Außerdem wird ein gemeinsames Netzwerk benötigt, in dem ein Broadcast über eine Broadcast IP-Adresse nicht gesperrt ist.

## Anforderungen

Das Programm soll eine ansprechende und für den Benutzer selbsterklärende Benutzeroberfläche haben.

Es soll die essentielle Funktion eines Chatprogramms erfüllen, nämlich das Versenden von Nachrichten über ein Netzwerk. Dabei sollen mehrere Chats parallel geführt werden können, sodass ein Wechsel zwischen mehreren Chatpartnern möglich ist. Die verschiedenen Chatpartner sollen in einer Kontaktliste organisiert sein.

Um die Chats auch nach einem neuen Öffnen weiterführen zu können, sollen die Kontakte und die dazugehörigen Nachrichten gespeichert werden.

Die Programmierung soll nachvollziehbar und gut strukturiert in Klassen aufgeteilt werden.

## Erweiterungen

Als mögliche Erweiterung des Programms können Gruppenchats eingeführt werden. Die in diesen Gruppenchat gesendeten Nachrichten werden jedem Gruppenmitglied angezeigt und jeder hat die Möglichkeit Nachrichten in diesen Chat zu senden.

Eine weitere Erweiterung ist das Verschicken von Anhängen, wie Bildern oder Dokumenten. Hierfür könnte eine Bytefolge statt dem Nachrichtentext übertragen werden. Dabei ist jedoch das Problem des Datenverlustes gravierend, da die ursprüngliche Datei nicht mehr zusammengesetzt werden kann, wenn ein Teil der Bytes fehlt.

Um dieses Problem zu lösen, kann als zusätzliche Erweiterung ein Handshake der Netzwerkteilnehmer eingefügt werden. Das bedeutet, dass ein Empfänger Rückmeldet, dass er eine Nachricht erhalten hat. Sendet er diese Rückmeldung nicht, muss die Nachricht erneut versendet werden.

## Genehmigung

|  |  |
| --- | --- |
| Datum: | 01.03.2019 |
| Unterschrift Auftraggeber |  |
| Unterschrift Projektteammitglied |  |
| Unterschrift Projektteammitglied |  |

## Anhang

* Pflichtenheft
* Flussdiagramm des Programmablaufs
* Technische Dokumentation
* Begriffserläuterungen
* Anwenderdokumentation

# Pflichtenheft

# Theoretischer Hintergrund

## IP-Adressen und Ports

Eine Internet Protokoll Adresse (IP-Adresse) ist ein eindeutiges Identifizierungsmerkmal und wird in einem Netzwerk für nur ein Gerät vergeben. Die meist verwendete IPv4 besteht aus vier Bytes.

Bsp: 192 . 168 . 0 . 2

Man unterscheidet bei der Vergabe von IP-Adressen zwei Verfahren. Bei der statischen IP-Vergabe wird einem Gerät eine feste Adresse zugeordnet, unter der dieses sich immer ins Internet einwählt. Im Gegensatz dazu steht die dynamische IP-Vergabe. Bei dieser Methode wird einem Gerät, das sich ins Netzwerk einwählen möchte, eine gerade freie IP-Adresse zugewiesen. Somit kann sich die IP-Adresse eines Geräts im Lauf der Zeit ändern.

Außerdem gibt es in einem Netzwerk spezielle IP-Adressen, die für einen Broadcast verwendet werden können, wie zum Beispiel 255.255.255.255.

Ein Port funktioniert als eine Art Schnittstelle zu einer höheren Ebene des Netzwerks. Dabei wird über einen Port das jeweilige Datenpaket weitergegeben. Portadressen liegen im Bereich von 0 bis 65535, wovon die Adressen 0 bis 49151 für spezielle Funktionen oder Programme, wie Email-Programme, reserviert sind. Die restlichen Ports von 49152 bis 65535 können dynamisch verwendet werden. Somit kann in einem Programm ein Port ausgewählt werden, über den alle Informationen ausgetauscht werden, die mit diesem Programm zusammenhängen. Dadurch werden auch nur diejenigen Datenpakete vom Programm ausgewertet, die auf dem ausgewählten Port eingehen.

## UDP-Protokoll

Das User Datagram Protocol (UDP-Protokoll) ist im Gegensatz zum Transmission Control Protokol (TCP-Protokoll) ein verbindungsloses Protokoll. Das bedeutet, dass es unsicher bei der Übertragung von Datenpaketen ist. Das bedeutet, dass der Sender eines Datenpaketes nicht weiß, ob dieses beim Empfänger angekommen ist.

Insgesamt ist es wesentlich schlanker gestaltet als TCP und enthält keinerlei Kontroll- und Datenmanagementfunktionen.

Genauso wie TCP unterstützt auch UDP die Kommunikation über Ports. Somit ist ein leichter Verbindungsaufbau möglich.

# Technische Dokumentation

## Die grundlegenden Funktionen des Programms

### Festlegen eines Usernames

Die essentielle Funktion eines Chatprogramms ist das Empfangen und Senden von Nachrichten. Es existieren viele unterschiedliche Protokolle, die das Aufbauen einer Kommunikation ermöglichen. Die Schwierigkeit bei einer solchen Netzwerkkommunikation ist, sicherzustellen, dass eine abgeschickte Nachricht auch bei dem Benutzer ankommt, an den sie geschickt wurde. In diesem Programm wird das UDP-Protokoll verwendet. Dabei handelt es sich um eine Peer-To-Peer-Kommunikation (Erklärung). Die Identifikation erfolgt über, vom User frei gewählte, Usernamen, die eine Länge von 3 bis 20 Stellen haben. Diese Methode ist nicht perfekt, da sie Doppelungen zulässt. Jedoch erlaubt sie einen gleichbleibenden Namen als Identifikation.

Folglich ist der Usernamen jedes Users essentiell für die Funktion des Programms und wird daher direkt beim ersten Start des Programms festgelegt. Die Eingabe erfolgt über einen benutzerdefinierten Dialog, der den eingebenden Namen auf die richtige Länge überprüft. Der gewählte Username wird nach der Eingabe gespeichert und kann vom User nicht mehr geändert werden. (siehe Speicherung)

### Senden einer Nachricht

Auf der Benutzeroberfläche (siehe Benutzeroberfläche) kann der User eine Nachricht eingeben und wahlweise über den Senden-Button oder das Drücken der ENTER-Taste an den momentan ausgewählten Kontakt schicken.

#### Die Klasse Message

Es wird jedoch nicht nur der Nachrichtentext versendet, sondern mehrere zusätzliche Informationen. Diese sind in einem Objekt „Message“ als Attribute zusammengefasst. Die Attribute sind:

* Sender
* Empfänger
* Zeitstempel
* Nachrichtentext
* Bool Sent, das den Abgeschickt-Status darstellt

Die Klasse enthält zusätzlich alle Methoden, die bei der Verarbeitung von Messages benötigt werden, wie beispielsweise das Codieren, bei dem alle obengenannten Informationen in einer bestimmten Reihenfolge zu einem String zusammengefügt und mit einer eindeutigen String-Zeichenfolge („$%&“) voneinander getrennt werden.

Das Attribut „Sent“ wird in unserem Programm bisher nicht verwendet. Es wurde für zahlreiche Erweiterungen des Programms vorgesehen. Zum Beispiel kann die Nachricht, wenn der Empfänger aktuell offline ist, zu einem späteren Zeitpunkt gesendet werden.

#### Senden der Nachricht

Zum Senden der Nachricht wird das UDP-Protokoll (Erklärung) verwendet. Zu diesem Protokoll stellt C# einen Udp-Client zur Verfügung, dem lediglich eine IP-Adresse und ein Port zugewiesen werden muss. In diesem Programm wird der Port 54546 verwendet. Dieser Port ist nicht für andere Zwecke reserviert. (Erklärung). Die Nachrichten werden an die IP-Adresse 255.255.255.255 gesendet. Werden Nachrichten an diese Broadcastadresse gesendet, erhalten sie alle Computer, die in diesem Netzwerk angemeldet sind. (Siehe Erklärung). Ist der Udp-Client initialisiert, kann über ihn ein Byte-Array verschickt werden, indem die Methode Send() der Udp-Clientklasse aufgerufen wird. Dieser Methode werden das Byte-Array und die Anzahl an bytes übergegeben.

In diesem Programm wird aus dem zu verschickenden String über die Funktion Encoding.Unicode.GetBytes(String) ein Byte-Array erzeugt, das dann verschickt wird.

### Empfangen einer Nachricht

Alle Computer eines Netzwerkes erhalten, die mit dem Chatprogramm verschickten Nachrichten. Jedoch werten nur die Computer die Nachrichten aus, die auch auf den entsprechenden Port achten. Normalerweise ignoriert ein Computer alle eingehenden Signale aus einem Netzwerk, die nicht auf einem der Standardports eingehen. Läuft dieses Chatprogramm, wird der Port 54546 auf dem die Nachrichten gesendet werden überwacht und die ankommenden Nachrichten ausgewertet.

#### Erhalten einer Nachricht

Beim Start des Programmes wird ein Thread (siehe Erklärung) gestartet, der parallel zum restlichen Programm arbeitet. Er überwacht den Port und ruft eine Methode auf, wenn eine Nachricht eingeht. Bei der eingehenden Nachricht handelt es sich um ein Byte-Array. Dieses wird zu einem String konvertiert, decodiert und in einem Messageobjekt gespeichert. Danach wird geprüft, ob die Nachricht für den User des Programmes bestimmt ist.

Sollte dies nicht der Fall sein, wird geprüft, ob der User selbst der Absender ist. Das ist notwendig da auch von einem selbst verschickte Nachrichten erst verarbeitet werden, wenn man sie selbst erhalten hat. Sollte der User weder Empfänger noch Sender sein, wird die Nachricht ignoriert.

##### Die Nachricht ist für den User bestimmt

Wurde die Nachricht an den User gesendet, erscheint ein Popup neben der Oberfläche, das den User über den Eingang der Nachricht informiert. Danach wird überprüft, ob der Sender der Nachricht in der Kontaktliste des Users vorhanden nicht. Ist dies der Fall, wird die Nachricht gespeichert (siehe Speicherung) und die Anzeige aktualisiert (siehe Oberfläche).

Ist der Sender nicht in der Kontaktliste des Users, wird ein Dialog aufgerufen. Nun muss der User entscheiden, ob der Sender als neuer Kontakt der Kontaktliste hinzugefügt werden soll. Entscheidet er sich dagegen, wird die Nachricht ignoriert. Entscheidet er sich dafür, wird der Sender der Kontraktliste hinzugefügt und die Nachricht gespeichert, sowie angezeigt.

##### Nachricht wurde vom User gesendet

Wurde die Nachricht vom User selbst versendet, wird sie ebenfalls gespeichert (siehe Speicherung) und auf der rechten Seite der Anzeige dargestellt. (siehe Oberfläche).

### Speicherung

Alle wichtigen Daten des Programms werden in einem XML-File gespeichert. Das XML-File wird beim Anlegen des eigenen Username erzeugt und mit dem Namen „Backupfile.xml“ im gleichen Ordner wie die .exe Datei gespeichert.

Die Datei ist in mehreren Nodes und Attributen organisiert (siehe Erklärung). Der Aufbau dieser Datei ist in Abbildung 1 zu sehen.

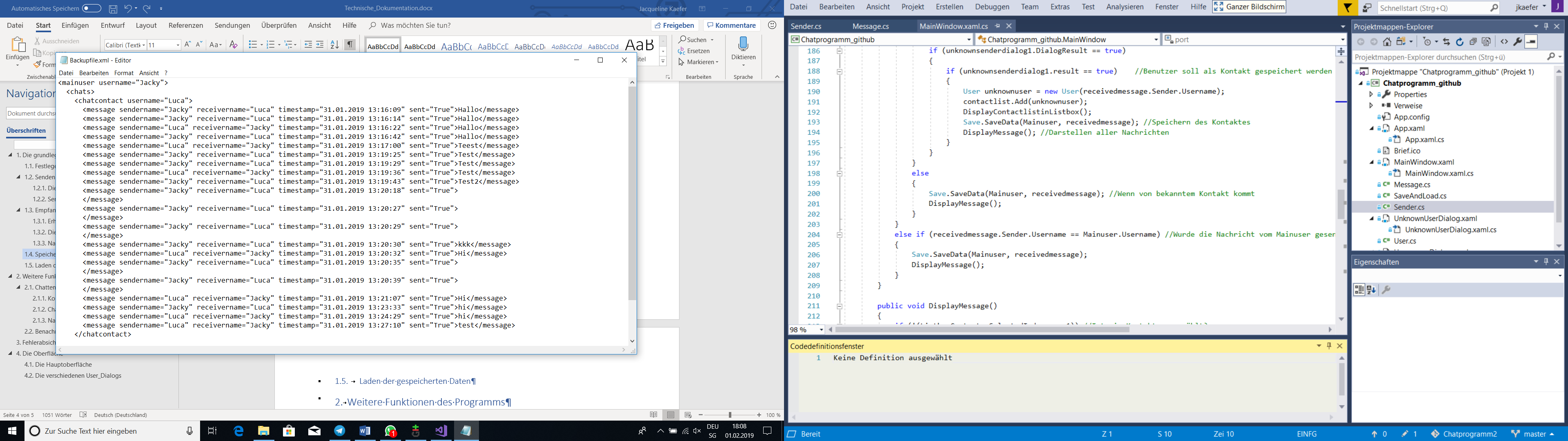


Abbildung 1: Muster-BackupFile

#### Speichern von Kontakten und Nachrichten

Die Datei enthält für jeden Chatkontakt eine chatcontact-node. An diese wird für jede Nachricht eine message-node angehängt Die Informationen zu den Kontakten und Nachrichten sind dabei in den Attributen gespeichert. Dies macht das Einfügen von neuen Kontakten und Nachrichten einfach, da lediglich eine neue Node mit den jeweiligen Attributen erzeugt und diese an die entsprechende Parent-Node angehängt wird.

#### Laden der gespeicherten Daten

Um Daten aus dem Backupfile zu laden, werden immer die entsprechenden Nodes in den Programmspeicher geladen. Anschließend können die gewünschten Informationen daraus ausgelesen werden. Beim Start des Programms wird beispielsweise der Username des Mainusers geladen, wenn bereits ein Backupfile besteht. Darüber hinaus werden alle Nachrichten des aktuell ausgewählten Chatkontakts geladen, wenn die Anzeige aktualisiert wird.

Die gewünschten Nodes können über die Funktion SelectNodes ausgewählt werden. Ihr wird eine Art Pfad übergeben, wie beispielsweise „//mainuser/chats/chatcontact“ zum Laden von Nachrichten. Die Funktion gibt eine Liste aller Chatkontakt-Nodes zurück. Über eine foreach-Schleife wird das Attribut „username“ mit dem aktuell ausgewählten Usernamen abgeglichen. Ist die richtige Chatkontakt-Node gefunden, können alle Nachrichten über die Funktion ChildNodes ausgewählt werden. Aus den Attributen dieser Nodes können die bisherigen Nachrichten geladen werden.

## Weitere Funktionen des Programms

### Chatten mit mehreren Personen parallel

Dieses Chatprogramm ermöglicht mit mehreren Personen parallel zu schreiben. Es kann zwischen den Chats gewechselt werden, indem der User auf den entsprechenden Kontakt in der Listbox klickt (siehe Oberfläche). Dabei kann immer nur ein Kontakt ausgewählt werden.

### Kontakte hinzufügen und löschen

Die Kontaktliste kann jederzeit um Kontakte erweitert werden. Dazu wird über den Button „Kontakt hinzufügen“ ein Dialog aufgerufen, in dem der Name des neuen Kontaktes eingegeben werden kann. Dieser Name muss dem Username des Kontaktes entsprechen.

Kontakte können über den Button „Kontakt löschen“ entfernt werden. Es wird jeweils der aktuell ausgewählte Kontakt gelöscht. Dadurch wird dieser aus der Kontaktliste entfernt und die entsprechende Chatkontakt-Node aus dem Backupfile, mit allen zugehörigen Nachrichten, gelöscht.

### Fehlerabsicherung

Mögliche Fehler des Programmes sind durch Try-Catch abgefangen. So zum Beispiel alle Methoden, die Daten im Backup-File speichern oder daraus auslesen. Aber auch der Thread der auf eingehende Nachrichten achtet, ist abgesichert, da auch hier Fehler auftreten können.

Bei manchen Fehlern, wie das Löschen des Backupfiles während das Programm läuft, wird es automatisch sicher geschlossen, da es ohne die Datei nicht zu verwenden ist. Beim nächsten Programmstart wird eine neue Datei erstellt.

## Die Oberfläche

### Die Hauptoberfläche

Die Hauptoberfläche des Programms enthält folgende Elemente von links nach rechts (siehe Abbildung 2):

* Eine Listbox zur Auswahl der Kontakte
* Zwei Buttons zum Löschen und Hinzufügen von Kontakten
* Ein Scrollviewer der ein Grid enthält, in dem die Nachrichten angezeigt werden
* Ein Button „senden“ zum Senden der Nachrichten
* Eine Textbox zur Eingabe von Nachrichtentexten

Die Listbox zeigt alle Elemente an die ihr als Items zugeordnet werden. Über die Funktion Items.Add werden ihr neue Items zugeordnet. Und mit der Funktion UpdateLayout werden sie dann auch angezeigt. Über das Event SelectionChangend kann eine Aktion ausgeführt werden, wenn der Nutzer einen anderen Kontakt auswählt. Dann wird in die Anzeige der Nachrichten aktualisiert, damit die Nachrichten des neu ausgewählten Kontaktes dargestellt werden.

Durch den Scrollviewer kann das darin enthaltene Grid größer als das eigentliche Fenser sein. Über einen Scrollbar kann trotzdem der gesamte Inhalt des Grid durchscrollt werden.

Dieses Grid hat zwei Spalten. In der linken Spalte werden eingehende Nachrichten dargestellt, in der rechten die Nachrichten, die der Users verschickt hat. Jede Nachricht wird in einer eigenen Zeile dargestellt. Die Höhe der jeweiligen Zeilen wird an die Länge des Textes angepasst, indem abgefragt wird, wie viele Zeilen die Textbox mit der Nachricht hat.

Der Button Senden löst mit dem Eventhandler ButtonClick das Senden des Textes in der Textbox unterhalb des Senden Buttons aus. Die Nachricht wird dann an den aktuell ausgewählten Kontakt gesendet.

Die Textbox dient der Eingabe des Nachrichtentextes.

Das Fenster der Anwendung kann nicht skaliert, sondern nur minimiert werden.

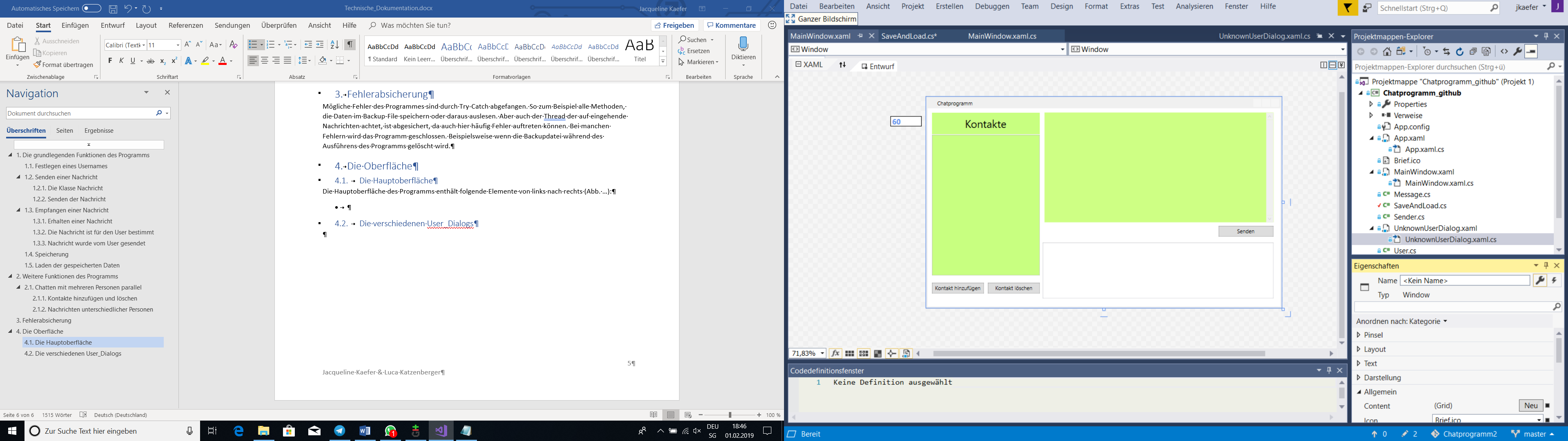


Abbildung 2: Oberfläche des Chatprogramms

### Die Benutzer-Dialoge

Dieses Programm verwendet zwei selbst programmierte Dialoge.

Der „Username\_Dialog“ dient zur Eingabe von Benutzernamen. Er enthält neben dem Ok-Button eine TextBox, zu Eingabe des Benutzernamens. Darüber hinaus sorgt er dafür, dass nur Usernamen angenommen werden, die den Anforderungen, wie Zeichenlänge, entsprechen.

Der UnknownUserDialog wird aufgerufen, wenn der Absender einer Nachricht nicht in der Kontaktliste gespeichert ist. Dieser Dialog enthält ein Label und zwei Buttons. Auf dem Label werden der Username des Absenders und die Frage, ob dieser gespeichert oder ignoriert werden soll, angezeigt. Die Entscheidung des Users wird über die Buttons „Speichern“ und „Ignorieren“ abgefragt.

# Anwenderdokumentation

Die Hauptfunktion, die das Programm bietet, ist wie der Name schon sagt, das Chatten mit anderen Kontakten. Im Folgenden ist beschrieben, wie der User das Programm verwendet.

## Starten des Chatprogramms

Wird das Programm zum ersten Mal gestartet, muss ein Username eingegeben werden. Dieser muss zwischen 3 und 20 Zeichen lang sein und kann im Nachhinein nicht mehr geändert werden. Bei allen folgenden Programmstarts werden alle Daten automatisch geladen.

## Kontakte hinzufügen und löschen

Der Benutzer hat zunächst die Möglichkeit über den Button „Kontakt hinzufügen“ einen neuen Kontakt in seiner Kontaktliste zu erstellen. Hierfür muss er lediglich dessen Benutzernamen in das angezeigte Dialogfeld eingeben. Die maximale Anzahl an Kontakten ist nicht beschränkt. Ein hinzugefügter Kontakt kann auch über den Button „Kontakt löschen“ wieder entfernt werden. Hierdurch werden jedoch auch alle bisher mit diesem Kontakt ausgetauschten Nachrichten unwiderruflich gelöscht.

## Nachrichten Senden

Es können nur Textnachrichten versendet werden, die jedoch nicht in der Zeichenanzahl beschränkt sind. Um eine Nachricht zu verschicken muss der Text zunächst in die große TextBox eingegeben und der Kontakt, an den die Nachricht gesendet werden soll, ausgewählt werden. Anschließend kann die Nachricht wahlweise über den Button „Senden“ oder das Drücken der ENTER-Taste abgeschickt werden. Anschließend wird sie im Nachrichtenverlauf oberhalb der TextBox dargestellt. Es ist nicht möglich Nachrichten zu verschicken, ohne dass ein Kontakt ausgewählt ist.

## Nachrichten empfangen

Erhält der Benutzer eine Nachricht, wird er durch einen Benachrichtigungston und ein Popup darauf hingewiesen. Dem Text des Popups kann er direkt entnehmen, von welchem seiner Kontakte die Nachricht gekommen ist. Ist dieser Kontakt gerade ausgewählt, wird die empfangene Nachricht auch sofort angezeigt.

Sollte der Sender einer Nachricht nicht in der Kontaktliste des Benutzers sein, erscheint ein Dialogfenster, das ihn auf die neue Nachricht hinweist. Der Benutzer kann sich daraufhin entscheiden, ob er den Sender, als neuen Kontakt, seiner Kontaktliste hinzufügen will, oder ob die Nachricht ignoriert werden soll.

## Speicherung

Empfange oder gesendete Nachrichten werden immer sofort in einem Backupfile gespeichert. Dadurch ist es möglich, dass mehrere Chats parallel geführt werden können und alte Nachrichten immer im Verlauf angezeigt werden. Darüber hinaus sind die Nachrichten auch nach dem Schließen und erneuten Öffnen des Programms noch verfügbar und werden nach dem Auswählen eines Kontakts im Verlauf dargestellt.

Alle Daten werden in einer Datei namens Backupfile gespeichert. Dieses sollte deshalb nicht gelöscht werden.

## Fehlerbehebung

Erscheinen während der Benutzung des Programms Fehlermeldungen, sollte der Benutzer den angezeigten Anweisungen folgen. Sollten unvorhergesehene Fehler auftreten, kann es helfen das Backupfile dennoch zu löschen, wodurch das Programm auf Werkseinstellungen zurückgesetzt wird. Das bedeutet auch, dass alle Nachrichten und Kontakte gelöscht werden.