Jacqueline Kaefer & Luca Katzenberger

Chatprogramm

Technische Dokumentation

Inhalt

[1. Die grundlegenden Funktionen des Programms 2](#_Toc536807454)

[1.1. Festlegen eines Usernames 2](#_Toc536807455)

[1.2. Senden einer Nachricht 2](#_Toc536807456)

[1.2.1. Die Klasse Nachricht 2](#_Toc536807457)

[1.2.2. Senden der Nachricht 2](#_Toc536807458)

[1.3. Empfangen einer Nachricht 3](#_Toc536807459)

[1.3.1. Erhalten einer Nachricht 3](#_Toc536807460)

[1.3.2. Die Nachricht ist für den User bestimmt 3](#_Toc536807461)

[1.3.3. Nachricht wurde vom User gesendet 3](#_Toc536807462)

[1.4. Speicherung 4](#_Toc536807463)

[1.5. Laden der gespeicherten Daten 4](#_Toc536807464)

[2. Weitere Funktionen des Programms 4](#_Toc536807465)

[2.1. Chatten mit mehreren Personen parallel 4](#_Toc536807466)

[2.1.1. Kontakte hinzufügen und löschen 4](#_Toc536807467)

[2.1.2. Chatpartner wechseln 4](#_Toc536807468)

[2.1.3. Nachrichten unterschiedlicher Personen 4](#_Toc536807469)

[2.2. Benachrichtigung über eine neue Nachricht 4](#_Toc536807470)

[3. Fehlerabsicherung 4](#_Toc536807471)

[4. Die Oberfläche 4](#_Toc536807472)

[4.1. Die Hauptoberfläche 4](#_Toc536807473)

[4.2. Die verschiedenen User\_Dialogs 4](#_Toc536807474)

# Die grundlegenden Funktionen des Programms

## Festlegen eines Usernames

Die essentielle Funktion eines Chatprogramms ist das Empfangen und verschicken von Nachrichten. Es existieren viele unterschiedliche Protokolle, die dies ermöglichen. Das Problem dabei ist, sicherzustellen, dass eine abgeschickte Nachricht auch dort ankommt, wo sie hinsoll und derjenige auch mitbekommt, dass diese Nachricht für ihn bestimmt ist. Es gibt viele Möglichkeiten, dies umzusetzen. In diesem Programm erfolgt die Identifikation über vom User freigewählte Usernamen in einer Länge zwischen 3 und 20 Stellen. Diese Methode ist nicht perfekt, aber sie erlaubt ein gleichbleibender Name als Identifikation und es ist kein Server erforderlich. Es handelt sich dabei um eine Peer-to-Peer-Kommunikation (Erklärung??).

Folglich ist der Usernamen jedes Users essentiell für die Funktion des Programms und beim ersten Start des Programmes muss als erstes ein Username festgelegt werden. Die Eingabe erfolgt über einen benutzerdefinierten Dialog, der den eingebenden Namen auf die richtige Länge überprüft und sofern er zwischen 3 und 20 Stellen lang ist, wird er an das Hauptprogramm zurückgegeben. Der gewählte Username wird nach der Eingabe gleich gespeichert. (Siehe Speicherung)

## Senden einer Nachricht

Auf der Benutzeroberfläche (Siehe Benutzeroberfläche) kann der User eine Nachricht eingeben und über den Senden-Button an den ausgewählten Kontakt schicken.

### Die Klasse Nachricht

Es wird jedoch nicht nur der Nachrichtentext versendet, sondern mehrere Informationen zusätzlich. Alle Informationen zu einer Nachricht sind in einem Objekt Nachricht als Attribute vorhanden. Die Attribute sind:

* Sender
* Empfänger
* Uhrzeit des Abschickens
* Nachrichtentext
* Ein Bool, der den Abgeschickt-Status darstellt.

All diese Informationen werden in einer bestimmten Reihenfolge zu einem String zusammengefügt und mit einer eindeutigen String-Zeichenfolge voneinander getrennt ($%&).

Die Klasse Nachricht wurde erstellt, um alle wichtigen Informationen zu einer Nachricht direkt in einem Objekt verknüpft zu haben. Außerdem fasst sie mehrere Methoden übersichtlich zusammen, die in Zusammenhang mit Nachricht benötigt werden. So zum Beispiel die Methode, aus einer Nachricht den oben erwähnten String herzustellen.

Das Attribut Bool wird in unserem Programm nicht verwendet, es kann aber für zahlreiche mögliche Erweiterungen des Programms verwendet werden. Zum Beispiel der Information, ob die Nachricht den Empfänger erreicht hat. Wenn nicht kann die Nachricht dann zu einem späteren Zeitpunkt nochmals abgeschickt 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 ein eine IP-Adresse und ein Port zugewiesen werden. In diesem Programm wird der Port 54546 verwendet. Dieser Port ist nicht für andere Zwecke reserviert. (Für eine Erklärung Siehe ….). Die Nachrichten werden an die Adresse 255.255.255.255 gesendet. Bei dieser Adresse handelt es sich um eine spezielle Adresse. Werden Nachrichten an diese Adresse gesendet, erhalten sie alle Computer, die in diesem Netzwerk angemeldet sind. (Siehe Erklärung….). Ist der Udp-Client initialisiert, kann ein Byte-Array damit verschickt werden, indem die Methode Send der Udp-Clientklasse aufgerufen wird. Dieser Methode wird der Byte-Array und die Länge des Arrays mitgegeben. 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 verschicken Nachrichten. Jedoch werten nur die Computer die Nachrichten aus, die auch auf den entsprechenden Port schauen. Normalerweise ignoriert ein Computer alle eingehenden Signale aus einem Netzwerk, außer sie geschehen auf einem der Standardports (Siehe Erklärung…). Läuft dieses Chatprogramm, wird der Port 54546 auf dem die Nachrichten gesendet werden überwacht und die einkommenden Nachrichten ausgewertet.

### Erhalten einer Nachricht

Beim Start des Programmes wird ein Thread (Siehe Erklärung….) gestartet, der parallel zum restlichen Programm, den Port überwacht und eine Methode aufruft, wenn eine Nachricht eingeht. Bei der eingehenden Nachricht handelt es sich um ein Byte-Array. Dieser wird als erstes zu einem String konvertiert und es wird ein Objekt Nachricht daraus erstellt. Die Klasse Nachricht enthält eine Methode dafür. Danach wird geprüft, ob die Nachricht überhaupt für den User des Programmes bestimmt ist. Es könnte sich auch um eine Nachricht handeln, die für einen anderen User bestimmt ist. Ist es nicht so, wird noch überprüft, ob die Nachricht vom User verschickt wurde. In diesem Programm schickt der Nutzer die Nachrichten, die er verschickt auch an sich selbst und diese werden auch vom Programm ausgewertet. Wenn der User weder der Empfänger noch der Sender ist. Wird nichts weiter mit der Nachricht gemacht.

### Die Nachricht ist für den User bestimmt

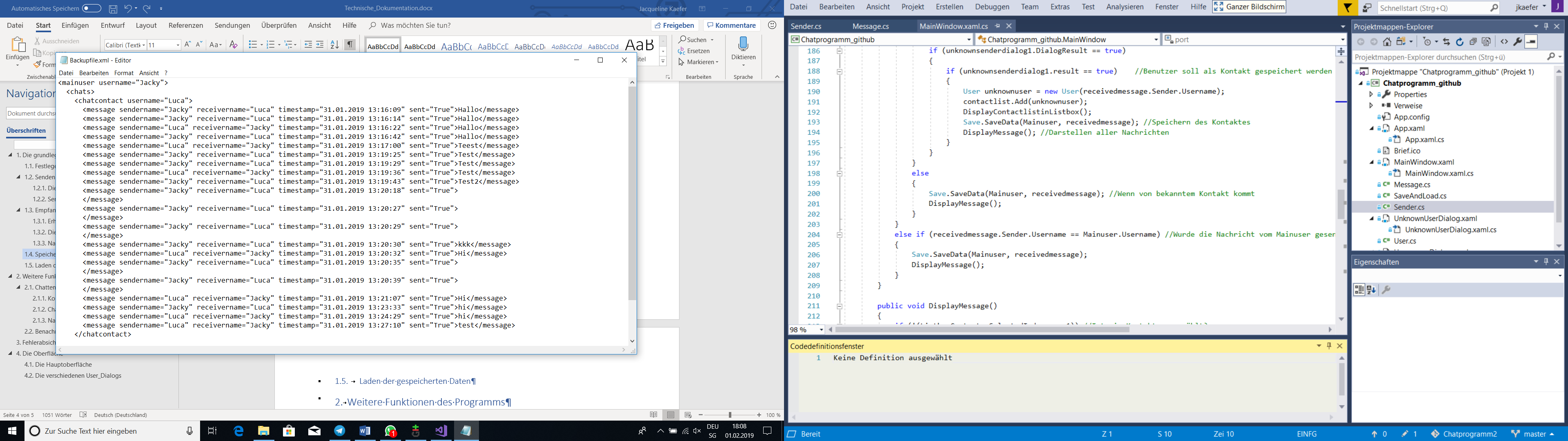
Wurde die Nachricht an den User gesendet, erscheint ein Popup neben der Oberfläche, das den User über den Eingang der Nachricht informiert. Danach wird überprüft, ob der Sender der Nachricht in der Kontaktliste des Users vorhanden nicht. Ist es so wird die Nachricht gespeichert (Siehe Speicherung) und die Anzeige wird aktualisiert (Siehe Oberfläche).Ist der Sender nicht in der Kontaktliste des Users, wird ein Dialog geöffnet, und der User muss entscheiden, ob der Sender als neuer Kontakt der Kontaktliste hinzugefügt werden soll oder nicht. Entscheidet er sich dagegen, wird nichts weiter mit der Nachricht gemacht. Entscheidet er sich dafür, wird der Sender der Kontraktilste hinzugefügt und die Nachricht wird gespeichert. (Siehe Speicherung) und die Anzeige wird aktualisiert (siehe Oberfläche).

### Nachricht wurde vom User gesendet

Wurde die Nachricht vom User versendet, wird sie ebenfalls gespeichert (Siehe Speicherung) und die Anzeige wird aktualisiert (Siehe Oberfläche).

## Speicherung

Alle wichtigen Daten des Programms werden in einem XML-File gespeichert. Das XML-File wird beim Anlegen eines neuen Usernames unter dem Namen Backupfile.xml im Bin-Ordner erzeugt. Die gespeicherten Daten sind in Nodes und Attributen organisiert Siehe Erklärung). Der Aufbau wird an einem Muster-Backupfile aufgezeigt.



Das File besteht aus einer Node Chatcontact für jeden Chatkontakt und einer Node Message angehängt an den Chatcontact für jede versendete Nachricht. Die Informationen zu den Kontakten und Nachrichten sind dabei in Attributen gespeichert. Dies macht das Einfügen von neuen Kontakten und Nachrichten einfach, da lediglich eine neue Node mit den Attributen erzeugt und diese an die entsprechende Parent-Node gehängt werden muss. Vor jeder Änderung muss das Dokument geladen und anschließend wieder gespeichert werden.

## Laden der gespeicherten Daten

Um Daten aus dem Backupfile zu laden, wird das ganze File in den Speicher geladen. Anschließend können die gewünschten Informationen geladen werden. Beim Start des Programmes wird beispielsweise der Username des Mainusers geladen, wenn bereits ein Backupfile besteht oder es werden alle Nachrichten des aktuell ausgewählten Chatkontakts geladen, wenn die Anzeige aktualisiert wird. Dies geschieht, wenn ein anderer Benutzer ausgewählt wird oder eine Nachricht eingeht.

Das Auslesen der Daten erfolgt über das Anwählen der gewünschten Node. Dies erfolgt über die Funktion SelectNodes dieser Funktion wird eine Art Pfad mitgegeben. Um Nachrichten zu laden zum Beispiel „//mainuser/chats/chatcontact“. Die Funktion gibt dann eine Liste aller ChatkontaktNodes zurück. Über eine foreach-Schleife werden dann die Attribute Username mit dem aktuell gewählten Usernamen abgeglichen. Ist die richtige Chatkontakt-Node gefunden, können dann alle Nachrichten über die Funktion ChildNodes ausgewählt werden. Aus den Attributen dieser Nodes können dann die Nachrichten geladen werden.

# Weitere Funktionen des Programms

## Chatten mit mehreren Personen parallel

Dieses Chatprogramm ermöglicht es mit mehreren Personen parallel zu schreiben. Es ist dabei immer nur einer angewählt und es kann zwischen den Chats gewechselt werden. Die Wahl des aktuellen Chatpartners erfolgt über eine ListBox (Siehe Oberfläche). Alle Kontakte sind in einer Kontaktliste gespeichert, die von der ListBox dargestellt wird.

### Kontakte hinzufügen und löschen

Die Kontaktliste kann jederzeit um Kontakte erweitert werden. Dazu wird über den Button Kontakt hinzufügen ein Dialog geöffnet, in der der Name des neuen Kontaktes eingegeben werden kann. Dieser Name muss dem Username des Kontaktes entsprechen. Ansonsten wird er die Nachrichten nicht erhalten, da die Identifikation auf den richtigen Usernamen angewiesen ist.

Kontakte können auch jederzeit über den Button Kontakt löschen entfernt werden. Es wird jeweils der aktuell ausgewählte Kontakt gelöscht. Dann wird der Kontakt aus der Kontaktliste entfernt und die entsprechende Chat-Kontakt-Node wird gelöscht zusammen mit allen zugehörigen Nachrichten.

### Nachrichten unterschiedlicher Personen

Die Nachrichten werden bei Empfang immer einem Chatkontakt zugeordnet und an entsprechender Stelle gespeichert. Auch wenn eine Nachricht eines nicht ausgewählten Chatkontakts eingeht, wird diese gespeichert und dargestellt, wenn er das nächste Mal ausgewählt wird.

# 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 häufig Fehler auftreten können. Bei manchen Fehlern wird das Programm geschlossen. Beispielsweise wenn die Backupdatei während des Ausführens des Programms gelöscht wird.

# Die Oberfläche

## Die Hauptoberfläche

Die Hauptoberfläche des Programms enthält folgende Elemente von links nach rechts (Abb. …):

1. Eine Listbox zur Auswahl der Kontakte
2. Zwei Buttons zum Löschen und Hinzufügen von Kontakten
3. Ein Scrollviewer der ein Grid enthält
4. In dem Grid werden Textboxen angeordnet, die die Nachrichten enthalten
5. Ein Button senden zum Senden der Nachrichten
6. Eine Textbox zur Eingabe von Nachrichtentexten

1) 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 den Event SelectionChangend kann eine Aktion ausgeführt werden, wenn der Nutzer einen anderen Kontakt auswählt. Dann wird in diesem Programm die Anzeige der Nachrichten aktualisiert, damit die Nachrichten des neu ausgewählten Kontaktes dargestellt werden.

2) Die zwei Buttons Kontakt löschen und hinzufügen lösen durch ihren Event ButtonClick aus, dass ein neuer Kontakt in einem Dialog eingegeben werden kann oder dass der ausgewählte Kontakt gelöscht wird.

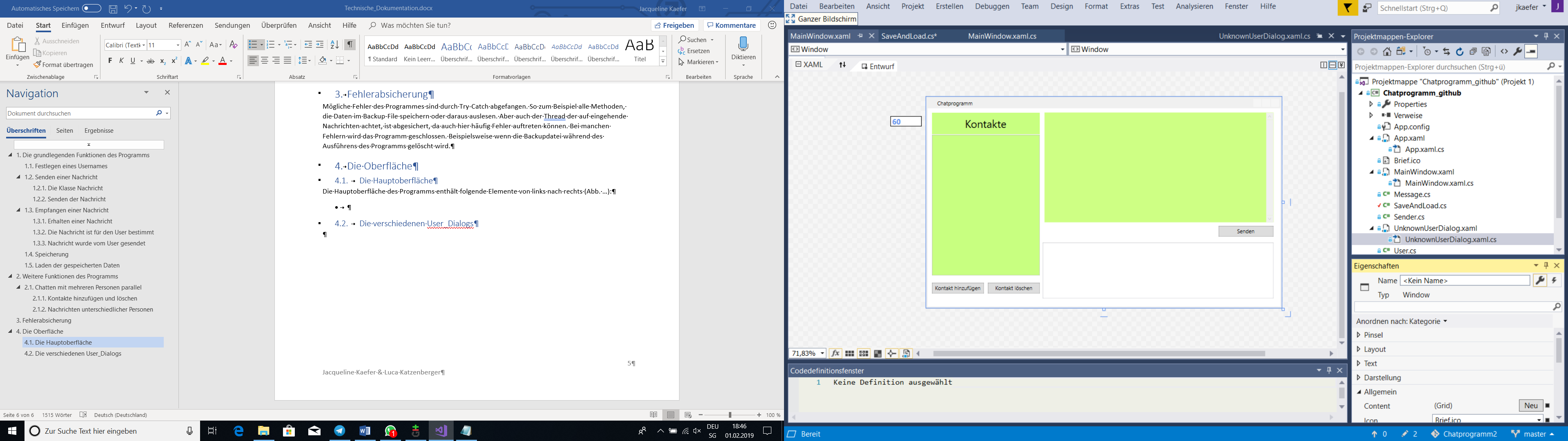
3) Der Scrollviewer ist dafür da, dass das darin enthaltene Grid grösser als das Fenser sein kann und trotzdem der ganze Inhalt durch scrollen über Scrollbars angeschaut werden kann. Ein Grid kann keine Scollbars haben. Und müsste in seiner vollen Grösse dargestellt werden, um den ganzen Inhalt anzeigen zu können.

4) Das Grid im Scrollviewer hat zwei Spalten. Und viele Zeilen. In der Spalte links werden Nachrichten vom Kontakt dargestellt und rechts werden die Nachrichten des Users dargestellt. Die Anzahl Zeilen entspricht der Anazhl Nachrichten. Jede Nachricht wird in einer Zeile dargestellt und die Höhen der Zeile wird der Länge des Textes angepasst, indem abgefragt wird, wie viele Zeilen die Textbox mit der Nachricht hat.

5) Der Button Senden löst mit dem Eventhandler ButtonClick das Senden des Textes in der Textbox unterhalb des Senden Buttons an den aktuell ausgewählten Chatkontakt aus.

6) Die Textbox dient der Eingabe des Nachrichtentextes.

Das Fenster der Anwendung kann weder vergrössert noch verkleinert werden. Die Grösse der Anzeigeelemente ist fest.



## Die verschiedenen Dialoge

Wenn der User bestimmte Daten eingeben muss, geschieht dies über extra Fenster, sogenannte Dialoge. Typische Dialoge sind beispielsweise der Print-Dialog oder der File-Dialog, die in C#-Bibliotheken enthalten sind. Es können jedoch auch Dialoge programmiert werden. Dieses Programm verwendet zwei selbst programmierte Dialoge. Ein Dialog dient der Eingabe von Benutzernamen. Er enthält neben dem Ok und Cancel-Button eine Textbox, zu Eingabe des Benutzernamens. Ausserdem gibt es einen Dialog zur Auswahl, ob ein unbekannter User der Kontaktliste hinzugefügt werden soll. Dieser Dialog enthält ein Label und zwei Buttons. Ein Button zum Speichern und zum Ignorieren des Users.