29.3.2015

Luca Tirabassi, Alex Theijs

Modulprüfung M120

Dokumentation und Eportfolio zum Chatprogramm

Inhaltsverzeichnis

[1 Aufgabenanalyse 2](#_Toc415415785)

[1.1 Beschreibung des Problems / System 2](#_Toc415415786)

[1.2 Use-Cases für zentrale Aufgaben 2](#_Toc415415787)

[2 Benutzeranalyse 2](#_Toc415415788)

[3 Ergonomieanalyse 2](#_Toc415415789)

[3.1 Grundsätze 2](#_Toc415415790)

# 1 Aufgabenanalyse

## 1.1 Beschreibung des Problems / System

Unser gewähltes Betriebssystem ist Windows. Weil wir uns für ein Chatprogramm entschieden haben, soll es dem Client möglich sein mit anderen Clients über einen Server miteinander zu kommunizieren. Auch der Server sollte ein Microsoft Windows Betriebssystem haben, da wir mit .NET arbeiten. Des Weiteren sollte eine Internetverbindung vorhanden sein, oder zumindest ein lauffähiges Netzwerk, damit die verschiedenen Clients miteinander kommunizieren können.

## 1.2 Use-Cases für zentrale Aufgaben

Folgende Usecases haben wir ausgearbeitet:

1. User startet Programm
2. User gibt in Textfeld seinen Namen ein
3. User loggt sich ein
4. User wählt Chatpartner aus
5. User wählt Chattextbox aus
6. User schreibt in TextBox
7. User sendet geschriebene Nachricht
8. User erhält geschriebene Antwort
9. User schliesst Applikation

# 2 Benutzeranalyse

Unsere Zielgruppe sind unerfahrene Computerbenutzer, welche einfach und schnell mit einer kurzen Installation und wenigen Klicks miteinander einfach und schnell kommunizieren wollen.

# 3 Ergonomieanalyse

## 3.1 Grundsätze

* Das Chatprogramm soll den Benutzer nicht überfordern und nur das können was wirklich von Ihm erwartet wird.
* Das Programm sollte möglichst selbsterklärend sein 🡪 Das bedeutet, dass der Benutzer ohne jegliches lesen einer Dokumentation mit dem Programm umgehen kann.
* Unser Programm soll ähnlich funktional und Benutzeroberflächen technisch gestaltet sein wie bereits vorhandene Produkte, wie zb.: Skype, ICQ, MSN, Viber
* Dem Benutzer soll es nicht möglich sein falsche Eingaben zu tätigen. Falls dies doch passieren sollte, soll dem Benutzer angezeigt werden, was er falsch gemacht hat.
* Die Bedienbarkeit des Programms soll schnell und einfach lernbar und ausführbar sein

# 4 Eportfolio

## 4.1 Was wollen wir lernen

Eine Server-Client Kommunikation über Sockets in C#.

## 4.3 Wieso wollen wir das lernen

Da wir uns beide stark für Spieleprogrammierung interessieren, dachten wir wäre es am besten, wenn wir uns in diesem Projekt intensiv die Server-Client Kommunikation anschauen.  
Dieses Wissen sollte uns als Grundlage für zukünftige Projekte dienen.

## 4.4 Wie haben wir es gelernt

In der Dokumentation von Microsoft sind gute Beispiele für C# Socket Programmierung vorhanden.  
Auch in den Java-Dokumentationen konnten wir andere Beispiele in einer ähnlichen Sprache vergleichen.  
Durch gewisse Code-Snippets und mit „herumspielen“ und korrigieren von Fehlern und veraltetem Code, kommt man schnell zum gewünschten Ergebnis.

## 4.5 Fazit

Es gibt viele verschiedene Vorgehensweisen, wie man einen Server oder einen Client programmiert. Je nach Anforderung müssen gewisse Punkte behandelt worden sein (implementiert), damit der Server oder der Client auch richtig kommunizieren können.

Für unser Beispiel haben wir einen ganz simplen Server programmiert, der aus relativ wenigen Zeilen besteht.  
Wie die Kommunikation miteinander funktioniert hat man nach dem Betrachten des Codes schnell verstanden.

# 5 Testing