Dokumentation zum Projekt *Dezentrales Chatprogramm*

Josef Schugt, Simon Seyock

2015-05-27

1 Zielbestimmung

Dezentrales Chatprogramm ist ein Windows-Programm, das den Echtzeitaustausch von Textnachrichten über das Internet zwischen Benutzern dieser Software ermöglicht. Diese Art der Kommunikation wird im weiteren Verlauf des Textes als *Chatten* bezeichnet.

Ein zentraler Server ist für die Kommunikation mittels *Dezentrales Chatprogramm* nicht erforderlich, soll aber in zukünftigen Versionen als optionales Mittel zum Auffinden von Kommunikationspartnern unterstützt werden.

2 Gliederung

Zunächst werden die Muss- und Sollkriterien für *Dezentrales Chatprogramm* dargestellt. Anschließend wird auf Designentscheidungen eingegangen, die sich nicht einem einzigen Kriterium zuordnen lassen, sondern von allgemeiner Natur sind. Den Abschluss des Dokuments bilden Abbildungen, die dem Verständnis der Software dienen, namentlich

- das Datenbankschema,
- das Klassendesign und
- ein Screenshot des zentralen Chatfensters.

3 Musskriterien

Die Musskriterien legen fest, welche Anforderungen die fertige Software zwingend erfüllen muss; bei *Dezentrales Chatprogramm* sind dies folgende:

1. Benutzer können sowohl mit einzelnen Benutzern als auch mit einer Gruppe von Benutzern chatten. Auf Netzwerkebene werden beide Chatarten mittels Nachrichten realisiert, die von je einem Sender an je einen Empfänger gesendet werden. Innerhalb der Software werden sie mit Hilfe von Konversationen abgebildet, zu denen zwei oder mehr Teilnehmer und eine beliebige Anzahl von Nachrichten gehören. Jede der Nachrichten ist genau einem Sender zugeordnet.

2. Benutzer können in einem Fenster chatten.

3. Benutzer können sich parallel an mehreren Chats beteiligen.

   Um eine Benutzerschnittstelle zu realisieren, die parallele Chats ermöglicht, werden Registerkarten verwendet. Als Alternativen zu diesen wurden ein Multiple Document Interface (MDI) und ein Single Document Interface (SDI) erwogen.

   Für ein MDI spricht, dass mehrere Chats *gleichzeitig* angezeigt werden können, was mit Registerkarten nicht möglich ist. Dem steht der Nachteil gegenüber, dass die Benutzeroberfläche bei einem MDI-Ansatz unübersichtlich wird. Dieser Nachteil war ausschlaggebend, von einem MDI abzusehen.

   Ein SDI-Ansatz anstelle von Registerkarten wurde ausgeschlossen, weil eine Möglichkeit zum Wechsel zwischen Chats benötigt wird und ein SDI hierfür keine intuitiv benutzbare Benutzerschnittstelle ermöglicht.

4. Der Chatverlauf wird gespeichert.

   Für das Speichern des Chatverlaufs wird eine lokale SQLite-Datenbank verwendet.

   Das wesentliche Argument für den Einsatz einer relationale Datenbank ist, dass sie alle für das Projekt notwendigen Möglichkeiten bietet und auf die zusätzlichen Möglichkeiten einer nicht-relationalen Datenbank verzichtet werden kann.

   Die Wahl einer relationalen Datenbank mit SQL als Datenbanksprache ergibt sich daraus, dass keine unnötige Einarbeitungszeit erforderlich

ist.

Für das lokale Speichern der Daten kommen grundsätzlich eine serverbasierte Lösung wie MySQL oder eine eingebettete Datenbank wie SQLite in Frage.

Gegen eine serverbasierte Lösung spricht, dass es dem Sinn eines dezentralen Chatprogramms widerspricht, vom Vorhandensein einer Serveranwendung abhängig zu sein. Außerdem erwarten Benutzer eines Chatprogramms eine möglichst einfache Installation, die sich bei einer serverbasierten Lösung nicht gewährleisten lässt.

Für eine eingebettete Datenbank spricht auch, dass bei der Implementierung keine Datenbankzugriffe anderer Programme berücksichtigt werden müssen.

5. Nachrichten können auch dann geschrieben werden, wenn der gewünschte Empfänger momentan nicht erreichbar ist.

   Noch nicht ausgelieferte Nachrichten werden ausgeliefert, sobald der gewünschte Empfänger (wieder) erreichbar wird. Dies ist durch die nächsten beiden Forderungen selbst dann gewährleistet, wenn *Dezentrales Chatprogramm* beendet und später neu gestartet wird.

6. Beim Programmende noch nicht gesendete Nachrichten werden automatisch gespeichert.

7. Stehen beim Programmstart Nachrichten aus einer vorherigen Sitzung zum Ausliefern an, werden diese genauso behandelt wie neu geschriebene.

8. Benutzer können andere Benutzer von der Kommunikation mit ihnen ausschließen, im Weiteren wird dieses Ausschließen *Blockieren* genannt.

9. Benutzer können ihre eigene Identität speichern.

10. Benutzer können eine Buddyliste führen, das ist eine gespeicherte Liste potentieller Chatpartner. Typischerweise sind dies Chatpartner, mit denen der Benutzer häufig kommuniziert bzw. kommunizieren will.

11. Nachrichten haben einen Zeitstempel, aus dem hervorgeht, wann sie abgesendet wurden. Dieser wird vom sendenden Chatprogramm gesetzt.

12. Eine Liste der offenen Chats wird bereitgestellt.

Diese Auflistung ist Teil der Funktionalität der Registerkarten des Chatfensters.

4 Sollkriterien

Die Sollkriterien legen fest, welche Eigenschaften der Software wünschenswert wären, jedoch nicht zwingend erforderlich sind, um die Vorgaben zu erfüllen.

1. Die Uhrzeit wird zwischen Instanzen von *Dezentrales Chatprogramm* synchronisiert.

   Dies verhindert die prinzipiell vorhandene Möglichkeit, dass Nachrichten nicht in der korrekten zeitlichen Reihenfolge angezeigt werden. Grund für dieses Problem ist, dass das Erstellungsdatum von unterschiedlichen Instanz des Chatprogramms gesetzt wird.

   Diskrepanzen in den Systemuhrzeiten der am Chat beteiligten Systeme führen ohne Synchronisation unter anderem dazu, dass Nachrichten in der falschen Reihenfolge angezeigt werden.

2. Popupbenachrichtigung ist möglich. Damit ist gemeint, dass die lokale Instanz von *Dezentrales Chatprogramm* Windows zu einer Benachrichtigung des Benutzers veranlassen kann.

   Ein typischer Anwendungsfall hierfür ist das Anzeigen einer Benachrichtigung darüber, dass der Name des Benutzers in einem Chat erwähnt wurde.

3. Text-Emoticons können in grafische Emoticons konvertiert werden. Hiermit ist gemeint, dass Textfolgen wie `:-)` oder `^_^` durch eine entsprechende Grafik ersetzt werden, in diesem Fall ein lächelndes Gesicht.

   Da diese Funktion je nach Inhalt der Kommunikation zu Verständnisproblemen führen kann, ist sie abschaltbar.

4. Die Kommunikation kann verschlüsselt erfolgen.

   Verschlüsseln der Nachrichten ermöglicht im günstigsten Fall:

   Geheimhaltung. Nur der gewünschte Empfänger kann die Nachricht lesen.

Authentifizierung. Der Empfänger kann überprüfen, dass die Nachricht vom vorgeblichen Absender stammt.

Integrität. Der Empfänger kann überprüfen, dass die Nachricht bei der Übertragung nicht verändert wurde.

Verbindlichkeit. Der Absender kann später nicht leugnen, dass er die Nachricht gesendet hat.

5 Designentscheidungen, die nicht einem einzigen Kriterium zugeordnet werden können

5.1 MVC-Entwurfsmuster

Dezentrales Chatprogramm folgt dem MVC-Entwurfsmuster, das heißt alle Programmteile sind eindeutig einem der drei Bereiche Datenmodel (Model), Anzeige (View) und Programmsteuerung (Controller) zugeordnet. Model und View sind hierbei voneinander unabhängig und kommunizieren lediglich über die Controller miteinander.

Aus folgenden Gründen wird das MVC-Entwurfsmusters verwendet:

Wiederverwendbarkeit. Das Model ist von der View unabhängig verwendbar (z.B. in einem Server)

Kapselung. Die anzeigespezifischen Windows-Forms-Klassen und -Methoden befinden sich in einem vom Rest der Software abgekapselten Bereich des Programms.

Strukturierung. Durch klare Zuordnung der Programmteile zu Model, View und Controller wird das Auffinden eines Programmteils innerhalb der Software vereinfacht.

5.2 Zentrale Ideen der Gliederung von *Dezentrales Chatprogramm*

1. Das Model repräsentiert die für das Chatprogramm wesentlichen Daten (Konversationen, Nachrichten, Benutzer).

Wenn die Daten geändert werden, wird den Controllern das Auftreten dieser Änderung mitgeteilt. Die Controller fordern die Anzeige dazu auf, sich entsprechend zu verändern.

Erfolgt in der Anzeige eine Aktion eines Benutzers, wird der Controller hierüber informiert. Er modifiziert gegebenenfalls das Model. Über diese Änderung werden die Controller wie vorher beschrieben informiert.

2. Logische Gruppierungen innerhalb der Anzeige werden zu eigenen Klassen zusammen gefasst, beispielsweise `BuddyListe`, `Chatfenster`, `TabControl` und `Loginfenster`.

3. Controller erzeugen gegebenenfalls entsprechende View- und Model-Objekte oder geben vorhandene Objekte an assoziierte Controller weiter.

5.3 Repository-Pattern

Für jede Model-Klasse, deren Instanzen in die Datenbank geschrieben werden, wird eine Repositoryklasse verwendet. Diese übernimmt das Zuordnen von Objekteigenschaften zu Datenbank-Tabellenspalten und umgekehrt. Die Repositoryklassen unterstützten grundsätzlich folgende Funktionalitäten:

- Objekte aus der Datenbank lesen
- Objekte in die Datenbank schreiben
- in der Datenbank vorhandene Objekte verändern
- vorhandene Objekte aus der Datenbank löschen

Gegebenenfalls benötigte komplexere Abfragen werden als Methoden der Repositoryklassen realisiert.

Die Repositoryklassen erhalten Verweise auf den Datenbankcontroller. Das MVC-Entwurfsmuster wird hier aufgeweicht, um den Repositorys den Zugriff auf die anderen Repositorys zu ermöglichen. Damit ist das Anfordern der mit dem Objekt assoziierten Objekte über die entsprechenden Repositorys möglich.

Indem die Repositorys die Assoziationen selbst erstellen, wird vermieden, dass alle Objekte im Anschluss außerhalb der Repositorys verknüpft werden

müssen. Dies wäre mit unangemessenem Aufwand verbunden.

5.4 Generische Datenbankklasse

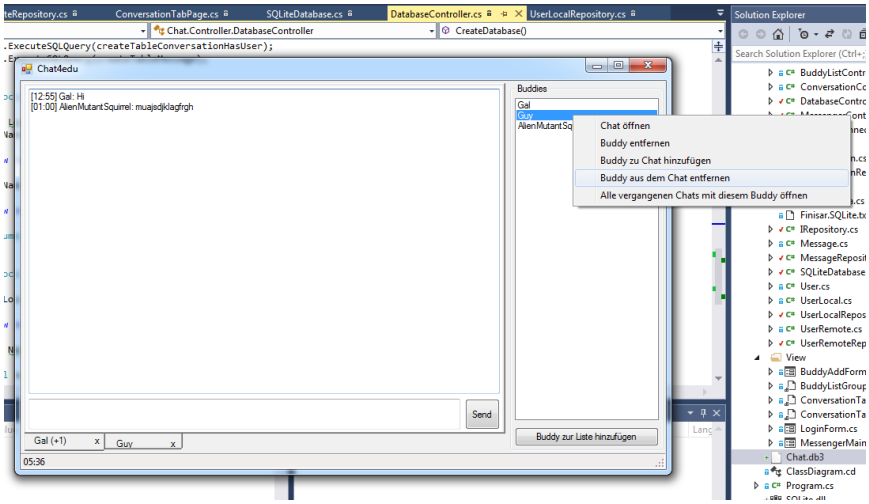
Um nicht jede Repositoryklasse für jedes Datenbanksystem neu schreiben zu müssen, verwenden die Repositoryklassen eine Datenbankklasse, die verallgemeinerte Methoden für den Zugriff auf Daten mittels SQL zu Verfügung stellt.

Für unterschiedliche Datenbanksysteme werden von dieser allgemeinen Klasse spezifische Klassen abgeleitet. In diesen werden die entsprechenden Methoden für ein konkretes Datenbanksystem implementieren.

6 Abbildungen

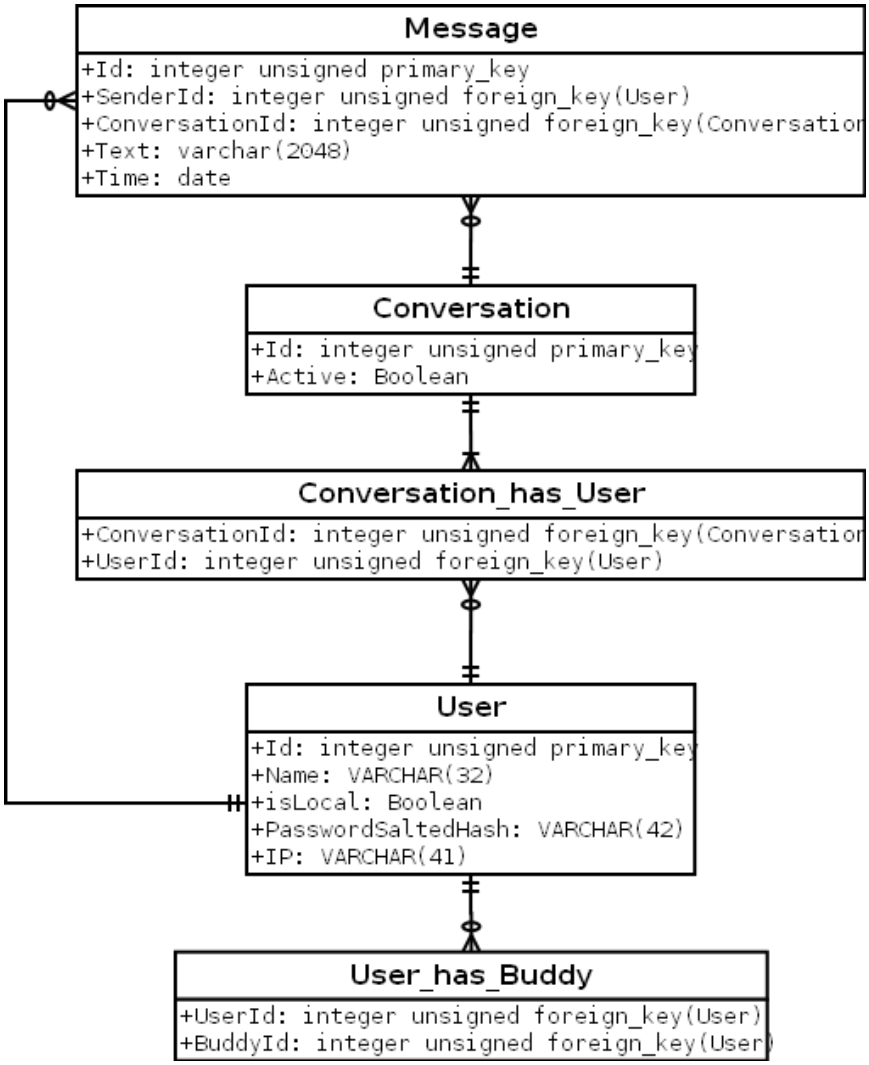
6.1 Screenshot

Der folgende Screenshot zeigt das zentrale Chatfenster von *Dezentrales Chatprogramm*. Anklicken der Abbildung öffnet die Grafik in Originalgröße.



6.2 Datenbankschema

Die folgende Abbildung zeigt das Datenbankschema von *Dezentrales Chatprogramm*. Anklicken der Abbildung öffnet die Grafik in Originalgröße.



6.3 Klassendesign

Die folgende Abbildung zeigt das Klassenschema von *Dezentrales Chatprogramm*. Anklicken der Abbildung öffnet die Grafik in Originalgröße.