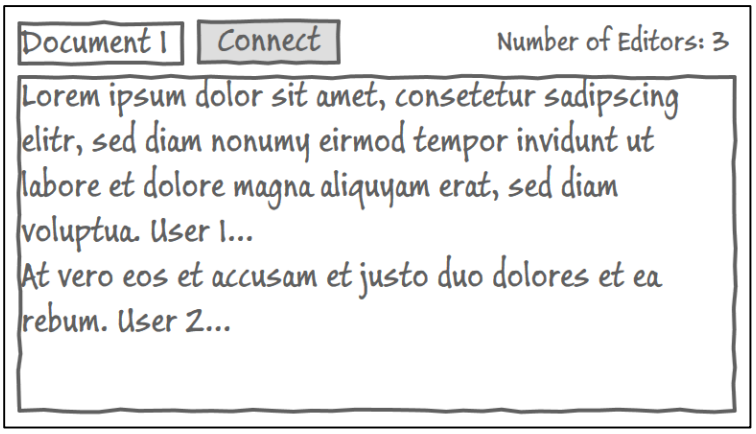
Übungsaufgabe: Shared Text Editor

Allgemeines

- Die Übungsaufgabe ist eine Gruppenaufgabe – bitte schließen Sie sich in Gruppen von **2 oder 3 Personen** zusammen (beachten Sie aber weiter unten angeführten zusätzlichen Anforderungen für eine 3-Personen Gruppe!).
- Die Abgabe erfolgt als ein einzelnes Zip File über **Moodle**. Zum Inhalt des Zip Files siehe unten.
- Deadline für die Abgabe ist der **21.12.2014 23:55 Uhr**.

Aufgabe

Bauen Sie einen Editor, mit dem mehrere Personen gemeinsam Text editieren können. Jede einzelne Person startet auf ihrem Rechner eine .Net-Anwendung, wo sie den aktuellen Zustand des Textes sehen kann, und diesen selbst und gleichzeitig mit anderen editieren kann. Die Anwendung soll ungefähr so aussehen:



Z.B. Während User 1 den ersten Satz editiert, könnte User 2 gleichzeitig den zweiten Satz editieren. Die Concurrency soll dabei so hoch wie möglich sein – selbst wenn 2 Personen Textstellen editieren, die nahe aneinander liegen, soll dies immer noch möglich sein. Das Editieren der exakt selben Textstelle, also quasi ein „übereinanderschreiben“, soll jedoch verhindert werden. Jegliche Synchronisierung soll so zeitnah wie möglich erfolgen (immer unter der Voraussetzung, dass die Stabilität der Anwendung und Integrität der Daten gewahrt ist) – idealerweise sollte man den anderen Usern also praktisch in Echtzeit beim Editieren der Daten zusehen können.

Das Herstellen der Verbindung soll über Eingabe des Dokument-Namens und drücken von „Connect“ erfolgen. Vorher soll es nicht möglich sein, Text zu editieren. Andere Personen sollen dann automatisch gefunden werden. Wenn man also z.B. „Document 1“ eingibt und dann „Connect“ drückt, und es editieren bereits 2 andere Personen ein Dokument mit diesem Namen, dann soll automatisch eine Verbindung zu diesen Personen hergestellt und der aktuelle Zustand synchronisiert werden.

Auch ein Verbindungsabbruch soll behandelt werden: Fällt die Verbindung zu einer Person aus (z.B. Netzwerkausfall oder Absturz der Anwendung), dann soll das Dokument nicht verloren gehen, sondern von den verbleibenden Personen weiter bearbeitetet werden können. Der aktuelle Zustand darf also nicht nur von einer Person allein gehostet werden.

Details zur Aufgabe

Wie Sie sich vermutlich schon gedacht haben, geht es bei dieser Aufgabe weniger um Parallelisierung, sondern vor allem um Concurrency, die Koordination des Zugriffs gemeinsame Datenstrukturen, sowie die Synchronisierung mehrerer Clients. Zu den wichtigsten Fragen, über die Sie sich Gedanken machen müssen, gehören daher:

- Wie sehen Ihre Datenstrukturen aus, so dass auf diese mit möglichst hoher Concurrency sowohl lesend als auch schreibend zugegriffen werden kann? Z.B. wird das Speichern des ganzen Textes in einer einzigen String-Variable nicht optimal sein, da dies bei absolut jeder Änderung einen Lock (und außerdem eine Neuerstellung des gesamten Strings) notwendig machen würde. Ihre Datenstrukturen sollten so entworfen sein, dass so viele Lese- und Schreibzugriffe wie möglich ohne Locks ausgeführt werden können.
- Wie werden Änderungen zwischen den Clients synchronisiert? Wie wird verhindert, dass gleichzeitige Änderungen bei verschiedenen Clients in unterschiedlicher Reihenfolge ankommen und damit möglicherweise bewirken, dass Zustände nicht mehr synchron sind?
- Wie führen Sie asynchrone Updates am Text in der GUI durch, ohne eine Person bei einer gerade stattfindenden Texteingabe zu stören (vor allem bei steigender Personenanzahl ein Problem)?
- Überlegen Sie, welche von .Net angebotenen asynchronen / Concurrency-Mechanismen Sie für die Aufgabe nutzen können (in der Vorlesung wurden einige vorgestellt, die hier nützlich sein können).

Weitere Details zur Implementierung:

- Bei der Implementierung sind Sie auf keine speziellen Technologien beschränkt. Die einzige Anforderung ist, dass Sie .Net (C#) und Visual Studio für die Aufgabe verwenden. Verwenden Sie gerne beliebige Frameworks zur Kommunikation oder auch für die GUI (zumindest solange diese Frameworks nicht die gerade genannten Problemstellungen bereits selbst lösen – denn dies soll ja Ihre Aufgabe sein – im Zweifelsfall fragen Sie nach, ob Sie ein bestimmtes Framework verwenden dürfen).
- Für die GUI können Sie WPF oder auch WinForms verwenden. Auch externe User Controls sind erlaubt. Die GUI muss nicht schön sein – wichtig ist, dass sie bedienbar ist und funktionell ihren Zweck erfüllt. Die oben gezeigt GUI Skizze ist ein Vorschlag dafür, was funktionell enthalten sein sollte – wenn Sie weitere Funktionen vorsehen wollen, steht Ihnen das natürlich offen.
- Zur Kommunikation können Sie z.B. WCF, XcoAppSpace, NServiceBus, Zyan, ZeroMQ oder MassTransit verwenden – die .Net Community stellt hier eine Unzahl an Möglichkeiten zur Verfügung.
- Überlegen Sie, wie Sie es bewerkstelligen können, dass sich die Clients beim Verbindungsaufbau gegenseitig finden. Es reicht, wenn dies innerhalb des gleichen Netzwerks funktioniert – z.B. mit einem UDP Broadcast. Verwenden Sie aber auch gerne Funktionen, die das Kommunikationsframework, das Sie verwenden, bereits zur Verfügung stellt.
- Persistenz ist keine Anforderung in dieser Aufgabe – d.h. es ist nicht notwendig, Dokumente über die Lebenszeit der Clients hinaus abzuspeichern. Will man ein bestehendes Dokument bearbeiten, kann man dessen Inhalt nach dem Start hineinkopieren (unterstützen Sie auf jeden Fall Copy/Paste).

Bericht

Zusätzlich zur Implementierung soll ein etwa 3-4 seitiger Bericht geschrieben werden, der zumindest folgendes beschreibt:

- Beschreiben Sie Ihre Implementierung. Welche Entscheidungen wurden getroffen und warum?
- Wie sieht ihre Lösung in Bezug auf Concurrency, Zugriffskoordination und Synchronisierung aus? Warum haben Sie sich für die gewählten Lösungsansätze entschieden? Welche verschiedenen Ansätze haben Sie ausprobiert und mit welchem Ergebnis?
- Für welches Kommunikationsframework haben Sie sich entschieden und warum?
- Wie viele gleichzeitige User schafft Ihre Lösung? Welche Probleme/Performanceeinbußen haben Sie dabei beobachtet?
- Auf welche Schwierigkeiten sind Sie gestoßen?
- Wieviel Zeit haben Sie wofür gebraucht?

3 Personen

Bei einer Gruppe mit 3 Personen untersuchen Sie außerdem den Datenverkehr, der zwischen den Clients stattfindet, sowie die Performance- und Speicherauslastung der einzelnen Clients. Wie wirkt sich eine Erhöhung der Anzahl der Clients auf diese Werte aus? Stellen Sie dies auch gern im Bericht grafisch dar. Beschreiben Sie weiters, welche Optimierungen Sie in dieser Hinsicht durchgeführt haben, und zu welchen Verbesserungen diese dann geführt haben.

Wie Sie die dafür benötigten Daten sammeln, bleibt Ihnen überlassen – z.B. wäre eine Erweiterung des Clients dafür denkbar, der die entsprechenden Daten in einem File aufzeichnet. Und/oder Sie verwenden die Profiling-Funktionen von Visual Studio oder andere Tools. Überlegen Sie auch, wie Sie wiederholbar gleiche Bedingungen z.B. bei der Frequenz der Texteingabe herstellen können, um wiederholbare Tests zu ermöglichen (simulierte Benutzereingabe?).

Abgabe

Im Zip File für die Abgabe sollen folgende Dinge enthalten sein:

- Visual Studio Solution mit der Lösung
- Jegliche benötigte externe Libraries, sofern diese nicht über NuGet automatisch heruntergeladen werden können.
- Zum Testen ein bat-File, mit dem 3 Instanzen des Editors (auf einem Rechner) gestartet werden können, die alle den gleichen Dokumentennamen voreingestellt haben (Sie müssen die vorkompilierte Anwendung aber nicht mit ausliefern – es reicht, wenn das bat-File nach dem kompilieren der Visual Studio Solution funktioniert).
- Bericht im PDF Format wie oben beschrieben