Dokumentation
zum Testat „Sampling Messages“

Vorlesung Verteilte Systeme, Kurs: TIT-15

Gruppe 6: Markus Müller
Sven Müller
Nicolai Hutschneider
Carmine Ippolito
Jona Neef
Janik Siller

Inhaltsverzeichnis

1 Motivation

Aufgabenstellung für das Testat ist die Erstellung eines Servers, welcher Dienste zum Erstellen, Löschen, Schreiben und Lesen von sogenannten Sampling Messages bereitstellt. Hiefür ist wichtig, dass die Messages über einen eindeutigen Namen identifiziert werden können. Der Server soll als Einzelinstanz eingesetzt werden, somit müssen Punkte wie Redundanz, Ortstransparenz, Migrationstransparenz, Relokationstransparenz und Replikationstransparenz vernachlässigt werden. Wichtig ist jedoch die Nebenläufigkeitstransparenz, d.h. dass der parallele Zugriff auf den Server von mehreren Instanzen möglich sein muss. Die Sicherheitsanforderungen sind reduziert, es wird keine Authentifizierung und Verschlüsselung zwischen Server und Client benötigt, es genügt eine Logging Funktion um alle Anfragen mit den entsprechenden Quelladressen zu protokolieren. Die Art der Umsetzung ist frei, es ist jedoch besonders auf Robustheit zu achten. Was die Sampling Messages betrifft, muss eine Namenslänge von mindestens 32 ASCII-Zeichen und eine Nachrichtenlänge von mindestens 255 ASCII-Zeichen garantiert sein. Es sollen mindestens 32 Nachrichten unterstützt werden, diese müssen aber nicht persistent gespeichert werden. Einschränkungen welche den verwendbaren Zeichensatz betreffen müssen dokumentiert werden. Mit der Aufgabenstellung wird ein Schnittstellenbeispiel mitgeliefert, wichtig ist, dass alle Anforderungen an die Schnittstelle, egal in welcher Form, umgesetzt werden.

Des weiteren soll ein Client erstellt werden, mit welchem alle vom Server angebotenen Dienste genutzt werden können. Hierfür gibt es keine weiteren Anforderungen mit Ausnahme davon, dass dem Endanwender die Antworten des Sampling Messages Server angezeigt werden sollen.

2 Umsetzung

Im Folgenden sollen Details zur Umsetzung von Server und Client genauer beleuchtet werden.

2.1 Architektur

Begründet durch bereits vorhandene Programmiererfahrungen soll das System für das Betriebssystem Windows 10 in der aktuellsten Version umgesetzt werden. Die Architektur ist schematisch in Abbildung 1 dargestellt. Der Sampling Messages Server soll als Einzelinstanz umgesetzt werden, der Client als Konsolenanwendung, welche parallel von mehreren Usern benutzt werden kann.

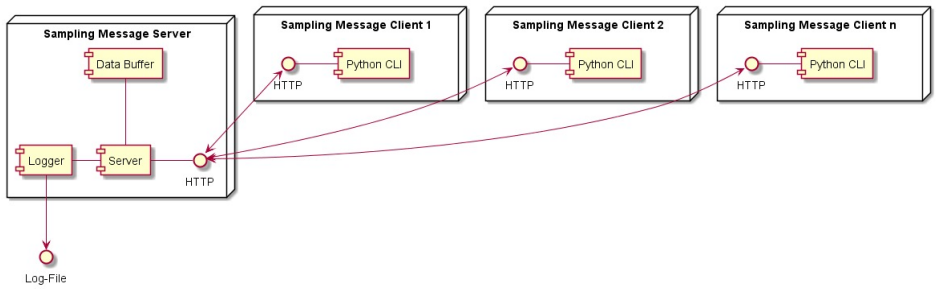

Abbildung 1: Architektur Sampling Messages Server

Der Server wird in C# , basierend auf dem .NET Framework in der Version 4.6.2 implementiert. Er bietet eine HTTP-Schnittstelle, die über eine RESTful API angepsrochen werden kann. Der Server stellt alle Dienste bereit und verarbeitet die von den Clients gesendeten Kommandos. Über ein Logger-Modul werden alle relevanten Informationen in ein Log-File gespeichert. Die nicht-persistente Speicherung der Sampling Messages wird über einen Data Buffer verwirklicht, auf welchen der Server zugreifen kann.

Der Client wird als Konsolenanwendung in Python umgesetzt. Er bietet die Möglichkeit, die RESTful API des Servers anzusprechen, wobei alle Konfigurationsmöglichkeiten über entsprechend übergebene Parameter abgedeckt werden.

2.2 Server

Der Server besteht aus den folgend gelisteten Klassen:

- ServerWindow
- Server
- HttpRequest

- HttpResponse
- Databuffer
- Log

In der ServerWindow Klasse werden alle Interaktionen mit der GUI verarbeitet. Dies umfasst das Starten und Stoppen, das Entgegennehmen der IP sowie das auslesen des Log-Verzeichnisses und die Ausgabe im Textfenster. Die Server Klasse initialisiert den Server nachdem der Start durch das GUI eingeleitet wurde. Sie lässt ihn als eigenen Thread laufen und übernimmt die Kommunikation mit den TCP-Clients. Ebenfalls ist die Klasse für das beenden des Servers verantwortlich. Mit der HttpRequest Klasse werden aus eingehenden Anfragen alle benötigten Daten herausgefiltert. Die HttpResponse Klasse baut dann auf Grundlage der Anfrage eine Antwort und sendet diese an den Client. Mit der Databuffer Klasse wird der Speicher gemanagt, d.h. übergebene Daten werden gespeichert und angefragte Daten ausgelesen. Ein möglicher Methodenaufruf innerhalb des Servers ergibt sich also wie folgt:

Listing 1: Exemplarischer Methodenaufruf

```
-> Start -> InitServer -> RunServer -> HandleTcpClient
-> GetRequest -> BuildResponse -> DataBuffer_Action -> SendResponse
```

Da die Nebenläufigkeitstransparenz für dieses verteilte System eine sehr wichtige Rolle spielt wird für jeden, sich verbindenden Client ein neuer Thread gestartet. Es werden also die Requests von allen Clients parallel abgearbeitet.

Als eindeutiger Identifier für die Sampling Messages dient ihr Name, die Sampling Messages werden immer darüber referenziert. Für die maximale Länge einer Sampling Message, deren Inhalt und die maximale Anzahl an speicherbaren Messages gibt es nur die Beschränkung, dass der Server pro Request maximal 1.000.000 Bytes empfangen kann.

Die Aktutalitätsinformation der Sampling Messages wird jeweils beim Aufruf von AddMessage, WriteMessage und ClearMessage aktualisiert. Soll eine Nachricht unendlich lang gültig sein muss die Aktualitätsinformation auf 0 gesetzt werden. In jedem anderen Fall wird die Aktualitätsinformation aus der aktuellen Zeit und dem übergebenen Gültigkeitswert in Sekunden berechnet.

Im Status jeder Sampling Message wird der letzte bekannte HTTP-Statuscode gespeichert, d.h. der Status kann alle verwendeten HTTP-Statuscodes als Wert annehmen.

2.3 Client

Um den Client zu realisieren und alle Anforderungen umzusetzen werden die folgenden drei Libraries verwendet:

- argparse
- http.client
- socket

Mit argparse werden die auf der Konsole eingegebenen Kommandos eingelesen und verarbeitet. Dies umfasst ebenfalls das parsen der übergebenen Parameter. Mit http.client werden die Verbindungen zum Server aufgebaut. Die Library socket dient als Erweiterung um die Exceptions der HTTP-Verbindungen genauer zu kategorisieren.

Um Robustheit auf der Seite des Clients abzusichern werden die übergebenen Parameter der Konsole mithilfe des Moduls argparse analysiert. Es werden nur korrekte Verkettungen angenommen und entsprechend an den Server weitergeleitet. Soweit möglich wird auch die Gültigkeit der Parameter (z.B. -v und –s) überprüft. Bei fehlerhaften Parametern und ungültigen Kombinationen wird dies dem User mit einer entsprechenden Fehlermeldung oder der Ausgabe der Hilfe angezeigt. Durch dieses Vorgehen erreichen den Server nur gültige HTTP-Requests.

Stimmen alle Parameter und deren Reihenfolge wird eine HTTP Verbindung zum Server aufgebaut. Die Verbindung wir dem Objekt vom Typ „SamplingMessageClient“mit allen Argumenten übergeben. Das Objekt dient zum verarbeiten der entsprechenden HTTP-Requests. Hierbei wird mit Ausnahme der GET und DELETE Requests jeweils ein JSON-String mit den entsprechenden Eingabeparametern erzeugt. Der String wird dann im Body über die HTTP-Verbindung an die RESTful API des Servers gesendet. Der Client wartet dann auf die Antwort des Servers. Erhält der Client die Antwort werden HTTP-Responsecode und HTTP-Responsereason ausgegeben. Bei manchen Requests wird zudem ein JSON-Body erwartert. Dieser wird dann geparsed und mit Responsecode und -reason ausgegeben. Ist die vom Server erhaltene Antwort falsch oder entspricht nicht der Erwartung wird dies dem User angezeigt. Mit dem anzeigen der Antwort endet die Aktion des Clients.

2.4 Schnittstelle

Die Kommunikationsschnittstelle des Servers ist eine HTTP-Schnittstelle, es wird somit das TCP Protokoll verwendet. Die Schnittstelle kann über eine RESTful API angesprochen werden. Die API sieht wie folgt aus.

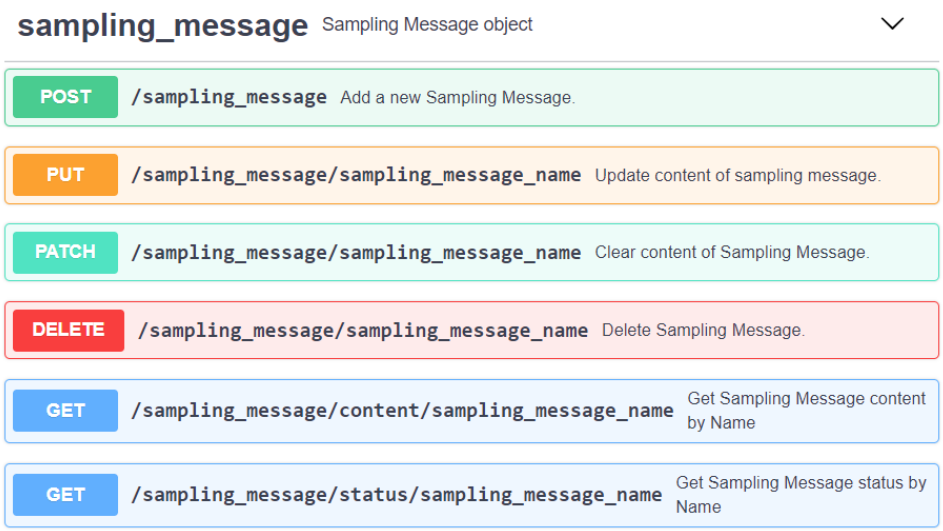

Abbildung 2: RESTful API der Kommunikationsschnittstelle

Für das Verwalten der Sampling Messages werden also die HTTP Methoden POST, PUT, PATCH, DELETE und GET verwendet. Für die Fehlerbehandlung werden die entsprechenden HTTP Fehlercodes verwendet.

3 Anwendungsbeschreibung

In diesem Kapitel soll beschrieben werden, welche Schritte zur Installation des gesamten Systems notwendig sind. Ebenfalls beschrieben werden soll, wie Server und Client bedient werden.

3.1 Installation

Um den Server zu starten wird das .NET Framework in der Verstion 4.6.2 benötigt. Falls das Programm nicht auf dem Computer vorhanden ist, kann dieses unter unten aufgeführtem Link [1] heruntergeladen werden. Ist die Installation abgeschlossen, kann der Server über die mitgelieferte, ausführbare Datei gestartet werden.

Für den Client muss Python in der Version 3.6 auf dem entsprechenden System installiert sein. Ist Python nicht installiert kann mithilfe des gelisteten Links [2] heruntergeladen werden. Wurde Python erfolgreich installiert, kann der Client über die Konsole aufgerufen werden

3.2 Bedienung Server

Beim starten des Sampling Messages Server wird die in Abbildung 3 dargestellte Applikation geöffnet. Die Benutzeroberfläche bietet die Möglichkeit, IP-Adresse und Port des Servers und den Dateipfad für das Log-File zu konfigurieren.

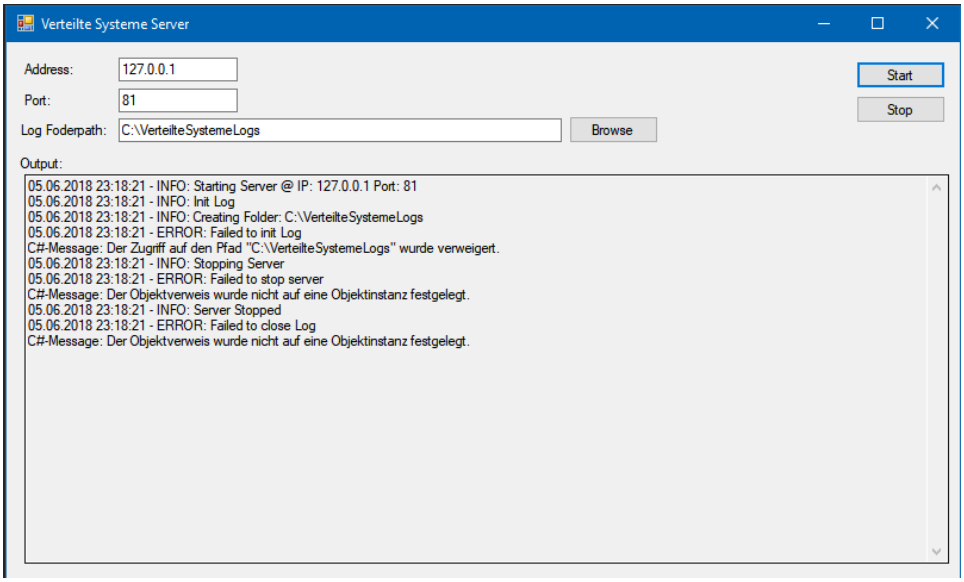

Abbildung 3: Benutzeroberfläche des Servers

[1] `https://www.microsoft.com/de-DE/download/details.aspx?id=56116`
[2] `https://www.python.org/downloads/`

Alle genannten Parameter sind frei konfigurierbar. Es ist jedoch darauf zu achten, dass sie nicht mit anderen, bereits im Netz laufenden Applikationen kollidieren. Über den Button „Start“kann der Server gestartet werden. Er öffnet die HTTP-Schnittstelle auf der definierten Adresse und nimmt Anfragen von Clients an. Die entsprechenden Ausgaben, Informationen und eventuelle Fehler werden in der Output-Textbox dokumentiert und ebenfalls im angegebenen Log-File permanent abgespeichert. Durch betätigen des Buttons „Stop “kann der Server beendet werden. Alle bis zu diesem Zeitpunkt gespeicherten Sampling Messages Dateien gehen dadurch verloren.

3.3 Bedienung Client

Der in Python realisierte Client kann über den nachfolgend dargestellten Konsolenaufruf ausgeführt werden. Zu beachten ist hierbei, dass der Pfad der Datei StandardInterface.py relativ angegeben werden muss. [dienst] steht in diesem Sinne für den gewünschten Dienst um entsprechende Messages zu Erstellen, Löschen, Schreiben oder Lesen. Die Adresse des Servers kann mit [–s IP:Port] definiert werden. Mit [-parameter] ist die entsprechend gewünschte Parameterliste gemeint. Ein Überblick über alle möglichen Parameter ist nachfolgend.

Listing 2: Konsolenaufruf für den Client

```
python StandardInterface.py --s 127.0.0.1:81 dienst -parameter
```

Zu Beginn empfiehlt es sich, die help Page des Clients auszugeben, um einen Überblick über die verschiedenen Dienste zu bekommen. Die Help Page kann über den nachfolgenden Konsolenbefehl aufgerufen werden.

Listing 3: Parameter für den Dienst create

```
python StandardInterface.py --s 127.0.0.1:81 -h
```

Der Client kann auf alle, vom Server angebotenen Dienste zugreifen. Die verschiedenen Dienste sind im Weiteren aufgelistet und werden genauer erklärt:

- create
- write
- clear
- read
- status
- delete

Mit dem Dienst create können neue Sampling Messages angelegt werden. Über den Parameter -n kann der für die Message gewünschte Name (String, z.B. sampleMessage1) definiert und mit dem Parameter -v eine entsprechende Validity Number (ganzzahliger Wert, z.B. 100) festgelegt werden. Die Parameter und ein exemplarischer Aufruf sind nachfolgend dargestellt.

Listing 4: Parameter für den Dienst create

```
-n MESSAGE_NAME -v VALIDITY_NUMBER
```

Listing 5: Aufruf des Dienstes create

```
C:\Users\Janik>python StandardInterface.py --s 127.0.0.1:81 create -n
    sampleMessage1 -v 100
Server response code: '200'
Server response reason: 'INFO: SampleMessage created successfull'
```

Mit dem Dienst write kann der Inhalt einer bereits bestehenden Sampling Message angepasst werden. Die Message wird über den Parameter -n mit ihrem eindeutigen Namen referenziert, der Inhalt (String, z.B. dies ist eine Sampling Message) kann über den Parameter -c übergeben werden. Die Parameter und ein exemplarischer Aufruf sind nachfolgend dargestellt.

Listing 6: Parameter für den Dienst write

```
-n MESSAGE_NAME -c MESSAGE_CONTENT
```

Listing 7: Aufruf des Dienstes write

```
C:\Users\Janik>python StandardInterface.py --s 127.0.0.1:81 write -n sampleMessage1
    -c dies ist eine sample Message
Server response code: '200'
Server response reason: 'INFO: SampleMessage updated successfull'
```

Mit dem Dienst clear kann der Inhalt einer bereits bestehenden Sampling Message gelöscht werden. Die Message wird über den Parameter -n mit ihrem eindeutigen Namen referenziert. Der Parameter und ein exemplarischer Aufruf sind nachfolgend dargestellt.

Listing 8: Parameter für den Dienst clear

```
-n MESSAGE_NAME
```

Listing 9: Aufruf des Dienstes clear

```
C:\Users\Janik>python StandardInterface.py --s 127.0.0.1:81 clear -n sampleMessage1
Server response code: '200'
Server response reason: 'INFO: SampleMessage cleard and marked as invalid'
```

Mit dem Dienst read kann der Inhalt einer bereits bestehenden Sampling Message gelesen werden. Die Message wird über den Parameter -n mit ih-

rem eindeutigen Namen referenziert. Der Parameter und ein exemplarischer Aufruf sind nachfolgend dargestellt.

Listing 10: Parameter für den Dienst read

```
-n MESSAGE_NAME
```

Listing 11: Aufruf des Dienstes read

```
C:\Users\Janik>python StandardInterface.py --s 127.0.0.1:81 read -n sampleMessage1
Server response code: '200'
Server response reason: 'INFO: SampleMessage readed successfull'
Message valid: 'True'
Message content: 'dies ist eine sample Message'
```

Mit dem Dienst status können die aktuellen Statusinformationen von bestehenden Sampling Messages abgerufen werden. Die gewünschte Message wird hierbei über den Parameter -n mit ihrem eindeutigen Namen referenziert. Der Parameter und ein exemplarischer Aufruf sind nachfolgend dargestellt.

Listing 12: Parameter für den Dienst status

```
-n MESSAGE_NAME
```

Listing 13: Aufruf des Dienstes status

```
C:\Users\Janik>python StandardInterface.py --s 127.0.0.1:81 status -n sampleMessage1
Server response code: '200'
Server response reason: 'INFO: SampleMessage readed successfull'
Message valid: 'True'
Message status: 'INFO: SampleMessage is not empty'
```

Mit dem Dienst delete kann eine bereits bestehende Sampling Message gelöscht werden. Die gewünschte Message wird über den Parameter -n mit ihrem eindeutigen Namen referenziert. Der Parameter und ein exemplarischer Aufruf sind nachfolgend dargestellt.

Listing 14: Parameter für den Dienst delete

```
-n MESSAGE_NAME
```

Listing 15: Aufruf des Dienstes delete

```
C:\Users\Janik>python StandardInterface.py --s 127.0.0.1:81 delete -n sampleMessage1
Server response code: '200'
Server response reason: 'INFO: SampleMessage deleted successfull'
```

Führt man alle exemplarischen Aufrufe nacheinander durch ergibt sich in der Konsole der in Abbildung 4 dargestellte Output.

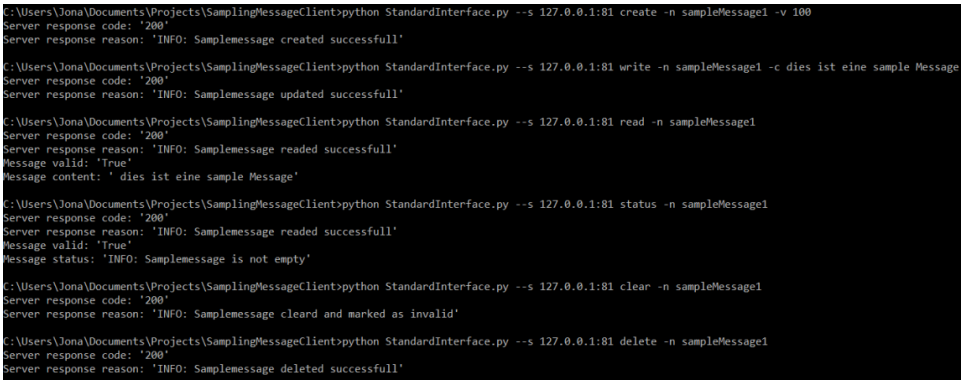

Abbildung 4: Beispielaufrufe