Verteilte Systeme Praktikum – Aufgabe 1

Team: 15, Adrian Helberg

Aufgabenaufteilung: 1er Team, keine Aufteilung

Quellenangaben:

- Aufgabe: http://users.informatik.haw-hamburg.de/~klauck/VerteilteSysteme/aufg1.html
- Entity-Relationship-Modell erstellt mit „Draw.io“ - https://www.draw.io/
- Informationen zur Erstellung eines Softwareentwurfs: http://www.enzyklopaedie-der-wirtschaftsinformatik.de/lexikon/is-management/Systementwicklung/Hauptaktivitaten-der-Systementwicklung/Softwareentwurf/index.html

Bearbeitungszeitraum: 05.10.2018 – 11.10.2018

Aktueller Stand:

- Entwurf
  - des Servers
  - der ADTs HBQ, DLQ und CMEM
  - des ER-Modells

Änderungen des Entwurfs: Version 1.0

Entwurf: Ab Seite 2

Protokoll

- „Nachricht des Tages“
  - Textzeilen
  - Verwaltungsinformation
    - Minimal 3, maximal 6 Elemente vom Typ Integer, String, 3-Tupel
- Log
  - Ausgaben
  - Fehler
  - Textdateien

Client

- Dienst
  - Sequentieller Wechsel zwischen Redakteur und Leser
  - Terminierung nach einer vorgegebenen Lebenszeit
- Redakteur:
  - Erstellen der „Nachrichten des Tages“
  - Anfordern von Nachrichten-IDs
- Leser:
  - Anmelden beim Server
  - Abfragen aktueller „Nachrichten des Tages“

Server

- Dienst
  - Bereitstellen von Nachrichten-IDs
  - Erweitern der empfangenen Textzeile um
    - die Empfangszeit
    - die Übertragungszeit
    - einen Zeitstempel
- CMEM
  - Lokal
  - Speichern von Lesern
- HBQ
  - Global
  - Hält Nachrichten, die nicht ausgeliefert werden dürfen
- DLQ
  - Lokal
  - Hält Nachrichten, die an die Leser ausgeliefert werden können

Zu implementieren ist die Server-Komponente mit den Sub-Komponenten CMEM, HBQ und DLQ

User Stories

- Epics
  - Lesern fragen die „Nachrichten des Tages“ ab
  - Redakteure verfassen die „Nachrichten des Tages“
  - Server verwaltet Anfragen für die „Nachrichten des Tages“

Funktionale Anforderungen

- Server
  - Basis-Strukturen Liste (lists) und Tupel (tuple) werden verwendet
  - Nachrichtenformat
    - MSG_List := [NNr,Msg,TSclientout,TShbqin,TSdlqin,TSdlqout]:
      - [Integer X String X 3-Tupel X 3-Tupel X 3-Tupel X 3-Tupel]
  - DLQ halt maximal ?Xdlq viele Nachrichten
  - Übertragungszeiten werden mit **erlang:timestamp()** getrackt
  - Sind keine Nachrichten beim Server vorhanden, wird eine Dummy-Nachricht versendet
  - Leser ohne Anfragen werden nach ?Xleser Sekunden beim Server abgemeldet
  - Besteht zwischen HBQ und DLQ zu 2/3 der Nachrichten eine Inkonsistenz, wird diese mit genau einer Fehlernachricht geschlossen
  - Nach einer gewissen Wartezeit ohne Anfragen, terminiert der Server
  - Die ADTs (HBQ, DLQ, CMEM) müssen austauschbar sein
  - Die HBQ wird als globaler ADT implementiert, DLQ und CMEM als lokaler
  - Konfigurationen wie Variablen (z.B. ?Xleser) werden in einer server.cfg – Datei angegeben
  - Ausgaben werden in Dateien Server<Node>.log und HB-DLQ<Node>.log geschrieben

Entity-Relationship-Modell