Verteilte Verarbeitung WS 18/19

Anbei eine Anleitung/ Vorgabe für das Fach "Verteilte Verarbeitung" im Wintersemester 2018/2019.

Allgemeines

Das Ziel von Verteilte Verarbeitung ist, dass der Student fortgeschrittene Programmierkenntnisse und eine selbständige Arbeitsweise / Problemlösungsfähigkeit entwickelt. Es wird drei Übungsaufgaben geben, die der Student selbständig bearbeiten muss. Die Übungsaufgaben besitzen jeweils eine Bearbeitungszeit von ca. 3 - 4 Wochen.
Ich bin für eure Unterstützung bei der Programmierung zuständig. Ich gebe euch Feedback zu eurem Code und gebe euch Hinweise für eine bessere / effizientere Programmierung. Wir werden in Einzelterminen bzw. Online euren Code gemeinsam reviewen und diskutieren. So soll ein Bewusstsein für guten Code geschaffen und eure Programmierkenntnisse gefestigt / verbessert weden.
Allerdings müsst ihr selbständig programmieren! Wenn es keinen Code beim Review gibt, kann ich euch nicht helfen.

Notengebung

Jede Übungsaufgabe wird von mir mit einer Note bewertet. Das Bewertungsschema ist pro Aufgabe individuell unterschiedlich, gliedert sich aber grob in folgende Bereiche:

- Funktionalität (doppelt gewichtet)
- Architektur (einfach gewichtet)
- Code Qualität (einfach gewichtet)
- Sonstiges (evtl. Bonuspunkte / je nach Aufgabe unterschiedlich)

Die 3 Übungsaufgaben sind alle gleich gewichtet. Es werden die 3 Noten der Aufgaben addiert und durch 3 geteilt.
Am Ende des Semesters wird es eine mündliche Prüfung geben, die von mir und Prof. Beneken geleitet wird. Diese Prüfung soll die berechnete Note der Übungsaufgaben bestätigen, kann allerdings auch von ihr abweichen. In der mündlichen Prüfung können Fragen zum Vorlesungsstoff aus Verteilte Verarbeitung vom Sommersemester 2018, aber auch Detailfragen zu eurem Code gestellt werden.

Kommunikation

Um eine einheitliche Kommunikationsplattform zu schaffen, erfolgt die Kommunikation über den Mattermost-Channel. Im Berufsumfeld wird auf diese Kommunikationsmittel gesetzt, wie beispielsweise Mattermost, Slack, Teams usw.
Teamname: VV_WS1819 / Channel: TownSquare
Bitte schaut regelmäßig in den Channel und seid generell aktiv im Mattermost! Hier wird kommuniziert und Neuigkeiten zu Übungen/Termine / Prüfungen verbreitet.

Allgemeine Fragen können im "Town Square"-Channel geklärt werden, aber auch private Nachrichten sind gerne willkommen.
Ihr dürft mich gerne jederzeit kontaktieren.

Terminvereinbarung

Die Terminvereinbarung erfolgt entweder:

- individuell über private Nachrichten im Mattermost
- über das Tool Slottr, anbei ein Demo Link vom letzten Jahr: http://www.slottr.com/sheets/10200982

Ich werde wöchentlich mögliche Terminvorschläge in die allgemeine Gruppe einstellen, hier könnt ihr euch für Termine eintragen. Ich bestätige anschließend den Termin, damit ihr wisst dass ich den Termin gesehen hab. Falls ihr keine Bestätigung für einen Termin erhaltet, sprecht mich bitte direkt im Mattermot darauf an.

Gitlab

Jeder Student besitzt ein eigenes Gitlab Repository für die Bearbeitung der Übungsaufgaben. Diese Repositories werde ich anlegen, wenn die passende Gruppe im Gitlab vorhanden ist.
Ihr müsst pro Aufgabe ein neuen Ordner im Repo anlegen. Am Ende dieses Semester wird es also 3 Ordner geben:

- 01_MealyAutomate
- 02_RestfulWebservice
- 03_Messaging

Achtet darauf, dass ihr eine passende .gitignore Datei anlegt !

Es wird mit Branches gearbeitet werden. Bitte verinnerlicht das Arbeiten mit Git, das ist sehr wichtig! Nicht nur für VV, sondern auch später im Berufsleben.
Der "master"-Branch ist schreibgeschützt, ihr könnt auf diesen also keine Commits pushen. Der "develop"-Branch ist der Entwicklungsbranch. Pro Aufgabe wird ein extra Branch ausgecheckt! Im jeweiligen Übungsbranch wird die Aufgabe bearbeitet und die Commits erstellt.

Online Feedback

Wenn ihr online Feedback zu eurem Code wünscht, erstellt ihr mir einen Merge Request und tragt mich als Reviewer ein. Das Arbeiten mit Merge bzw. Pull Requests ist in der Arbeit Best Practice und Code Reviews sind ein Zeichen von guter Arbeitsweise und eine Möglichkeit Codequalität sicherzustellen.

Ich werde Anmerkungen zu eurem Code direkt erstellen, die ihr dann verbessern könnt. Bitte checkt nach Erstellung eines Merge Requests regelmäßig euer Repo um meine Änderungen zu sehen. Alternativ stellt im Gitlab die Benachrichtigung für Änderungen an eurem Repository.

Nutzt die Möglichkeit online Feedback zu erfragen, nur so kann ich euch helfen und ihr euch verbessern! Häufige Merge Requests können euch helfen, dass ihr nicht in die "falsche" Richtung bei euren Projekten lauft und anschließend ein Problem bei der mündlichen Prüfung besitzt.

Ein Merge Request sollte immer folgendes Schema besitzen:

- Übungsbranch_Name → develop
- Reviewer: Thomas Mildner

Anbei eine Anleitung für Merge Requests: https://www.youtube.com/watch?v=Ddd3dbl4-2w

Achtung: Diese Anleitung ist noch für eine alte Gitlab Version - die Oberflächen unterscheiden sich ein wenig.
Prof. Beneken hat eine Anleitung für den Gitlab Server, der FH Rosenheim erstellt.
Zu finden in meinem Gitlab Repository: http://bit.ly/GitlabMergeRequests

Ich freue mich auf die Zusammenarbeit und ein produktives Semester!
Thomas